Representing and Learning Linguistic Structures on the Conceptual, Morpho-Syntactic and Semantic Level

Proefschrift voorgelegd tot het behalen van de graad van doctor in de wetenschappen aan de Vrije Universiteit Brussel te verdedigen door

Jens NEVENS

Promotoren: Brussel, 19 oktober 2022
Prof. dr. Katrien Beuls
Dr. Paul Van Eecke
Prof. dr. Ann Nowé
# Jury Members

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chairman</strong></td>
<td>Prof. dr. Wolfgang De Meuter</td>
<td>Vrije Universiteit Brussel, Belgium</td>
</tr>
<tr>
<td><strong>Secretary</strong></td>
<td>Dr. Roxana Rădulescu</td>
<td>Vrije Universiteit Brussel, Belgium</td>
</tr>
<tr>
<td><strong>External members</strong></td>
<td>Prof. dr. Wim Vanhoof</td>
<td>Université de Namur, Belgium</td>
</tr>
<tr>
<td></td>
<td>Dr. Remi van Trijp</td>
<td>Sony Computer Science Laboratories Paris, France</td>
</tr>
<tr>
<td></td>
<td>Prof. dr. Tias Guns</td>
<td>Katholieke Universiteit Leuven, Belgium</td>
</tr>
<tr>
<td><strong>Promotors</strong></td>
<td>Prof. dr. Katrien Beuls</td>
<td>Université de Namur, Belgium</td>
</tr>
<tr>
<td></td>
<td>Dr. Paul Van Eecke</td>
<td>Vrije Universiteit Brussel, Belgium</td>
</tr>
<tr>
<td></td>
<td>Prof. dr. Ann Nowé</td>
<td>Vrije Universiteit Brussel, Belgium</td>
</tr>
<tr>
<td><strong>Internal member</strong></td>
<td>Prof. dr. Laura Rosseel</td>
<td>Vrije Universiteit Brussel, Belgium</td>
</tr>
</tbody>
</table>
Language is a unique hallmark of human intelligence. Our linguistic systems do not only meticulously serve our communicative needs, they are also incredibly robust to noise, adaptive to change and they can be learnt efficiently. One of the grand challenges in AI consists in the development of truly intelligent autonomous agents with bidirectional communication systems offering the same robustness, adaptivity and efficiency as found in human languages. Ideally, these communication systems are also transparent and explainable in human-interpretable terms, which facilitates the verification of their internal consistency, thereby eliciting trust. Today, the design of adequate representations and learning mechanisms that lead to such communication systems is still an unsolved problem. In this dissertation, I present novel representations and learning mechanisms that are suitable for the communication systems of autonomous agents and that exhibit the key desirable properties of human languages. These representations and learning mechanisms are validated through case studies that tackle visual question answering benchmarks. Visual question answering involves answering natural language questions about images and requires perceptual, linguistic, and reasoning abilities. I present four main contributions that allow an agent to represent and acquire linguistic structures on the conceptual, morpho-syntactic, and semantic level. The first contribution consists in a computational construction grammar that has sufficient representational capacity for the benchmark dataset. Specifically, it provides bidirectional mappings between all questions of the dataset and their underlying procedural semantic representations. The next contributions consist of experiments where the agent acquires this grammar through task-based communicative interactions. Concretely, in the first experiment, the agent learns to extract meaningful concepts from continuous sensorimotor observations. Here, I build further on experiments within the language game paradigm, and I introduce a novel concept representation that is based on prototype theory. The second experiment provides an alternative, hybrid approach, which combines symbolic and sub-symbolic techniques to capture the same concepts in the form of hybrid procedural semantic representations. In the final experiment, I introduce a mechanistic model of two cognitive processes that are central in usage-based theories of language acquisition: intention reading and pattern finding. Through these processes, the agent is able to acquire a grammar through task-based communicative inter-
actions. In particular, the intention reading process enables the agent to hypothesize about the intended meaning of an observed utterance, and the pattern finding process enables the agent to construct abstract schemata that capture generalisations over form-meaning mappings. Together, the contributions that I present in this dissertation push forward the state of the art in the development of autonomous agents with communication systems offering human-like properties. The methodological advances that I introduce are relevant to a wide range of application domains, including for example human-robot interaction systems, conversational agents and intelligent tutoring systems.
Samenvatting

Taal is een uniek kenmerk van menselijke intelligentie. Ons taalsysteem is niet enkel uitermate geschikt voor onze communicatieve noden, het is ook zeer robust, flexibel en kan efficiënt geleerd worden. Een van de grote uitdagingen binnen de Artificiële Intelligentie bestaat uit het ontwikkelen van intelligente autonome agenten met bidirectionele communicatiesystemen die dezelfde robuustheid, flexibiliteit en efficiëntie als menselijke talen bezitten. Idealiter zijn deze communicatiesystemen ook transparant en verklaarbaar in menselijke termen, aangezien dit de interpretatie en verificatie van de interne werking van deze systemen vergemakkelijkt en daardoor vertrouwen uitlokt bij mensen. Vandaag is het ontwerpen van representaties en leermechanismen die leiden tot zulke communicatiesystemen nog steeds een onopgelost probleem. In deze doctoraatsthesis presenteer ik nieuwe representaties en leermechanismen die geschikt zijn voor de communicatiesystemen van autonomone agenten en de gewenste eigenschappen van menselijke talen tentoonstellen. Deze representaties en leermechanismen worden gevalideerd door casestudy’s die de ‘visual question answering’ taak aanpakken. ‘Visual question answering’ omvat het beantwoorden van vragen, gesteld in natuurlijke taal, over afbeeldingen en vereist perceptie, linguïstische vaardigheden en redenering. Ik presenteer vier bijdragen die een agent toelaten om linguïstische structuren op het conceptuele, morfosyntactische en semantische niveau voor te stellen en te leren. De eerste bijdrage bestaat uit een computacionele constructiegrammatica die voldoende representatieve capaciteit heeft voor de dataset van de casestudy. Specifiek voorziet deze grammatica bidirectionele mappings tussen alle vragen uit de dataset en hun onderliggende procedurele semantische representatie. De andere bijdragen bestaan uit experimenten waarbij de agent deze grammatica leert door middel van taak-gebaseerde communicatieve interacties. In het eerste experiment leert de agent om betekenisvolle concepten te abstraheren uit continue sensomotorische observaties. Hiervoor bouw ik verder op eerdere experimenten binnen het taalspel-paradigma en introduceer ik een nieuwe representatie voor concepten die gebaseerd is op prototype theorie. Het tweede experiment bestaat uit een alternatieve hybride aanpak voor het leren van concepten, waarbij symbolische en subsymbolische technieken gecombineerd worden om concepten te vatten in de vorm van een hybride procedurele semantische representatie. In het laatste experiment introduceer ik een mechanistisch model van twee cognitieve pro-
cessen die centraal staan in gebruiksgbaseerde theorieën van taalverwerving: ‘intention reading’ en ‘pattern finding’. Door middel van deze processen kan de agent een grammatica leren via taak-gbaseerde communicatieve interacties. In het bijzonder laat ‘intention reading’ de agent toe om hypotheses te maken over de bedoelde betekenis van een geobserveerde uitdrukking en laat ‘pattern finding’ de agent toe om abstracte schema’s te construeren die generalisaties over mappings tussen vorm en betekenis vatten. De bijdragen die ik in deze doctoraatsthesis presenteer stuwen de state-of-the-art in de ontwikkeling van autonome agenten met communicatiesystemen met mensachtige eigenschappen vooruit. De methodologische vernieuwingen die ik introduceer zijn relevant voor een brede waaier aan applicatiedomeinen, zoals mens-robot interactiesystemen, conversationele agenten en intelligente leermeestersystemen.
Acknowledgements

There are many people I wish to express my gratitude to for their role in the realisation of my dissertation. First of all, I would like to thank my supervisors Katrien Beuls and Paul Van Eecke. Thank you for your guidance, your invaluable feedback and advice, the opportunities you have given me and the trust you have placed in me. I am honoured to have witnessed first-hand how our research group has grown from the three of us at the old AI Lab in building G to a group of ten across three universities. This is due to the lively research environment with an ambitious and inspiring research programme that you have created. You have taught me many things about science, research, academia, teaching and life, including that doing the impossible is very well possible and that creating whole new scientific disciplines can happen over the course of a single experiment. I cannot wait to see what the future holds.

I am also indebted to the members of my PhD jury: prof. dr. Wolfgang De Meuter, dr. Roxana Rădulescu, prof. dr. Laura Rosseel, prof. dr. Tias Guns, dr. Remi van Trijp and prof. dr. Wim Vanhoof. Thank you for the time that you invested in assessing my manuscript and for your thoughtful comments and suggestions. They have greatly improved the quality of my work.

My gratitude also goes out to my colleagues Lara Verheyen, Jérôme Botoko Ekila, Jonas Doumen, Tom Willaert, Veronica Schmalz, Jeroen Van Soest, and Liesbet De Vos. Thank you for all the interesting discussions about my work and challenging me with your questions, but also for making every conference and offsite meeting an unforgettable experience and all the fun moments during hikes and frisbee matches. I wear the FCG sports jersey with pride because of all of you. Special thanks to Lara Verheyen for sharing the office with me the past three years, and for all the serious and many more less serious discussions. Your boundless curiosity never fails to improve my work. Thanks to Luc Steels, the scientific father of many of the ideas that make up the foundations of this dissertation, for your precious advice and inspiring talks. Thanks to Ann Nowé for your feedback as member of my advisory committee and for taking over the torch as ‘promotor-woordvoerder’ in the final months of my PhD. Thanks to Frederik Himpe for the technical support, and to Brigitte Beyens, Lara Mennes and Sien De Graeve for the administrative support. I would
also like to thank all other members of the AI Lab for the interesting discussions at research meetings, for helping me prepare my FWO defence and for many amusing lunch breaks, team building events, receptions and other lab activities.

I want to thank my parents for always giving me the freedom to pursue my dreams, together with my sister and my family-in-law for their continuous support throughout my education and scientific career. I am very fortunate to have two warm homes where I can always go in need of advice or to clear my mind.

Finally, I want to thank the most important person in my life. Thank you, Hanne, for your continuous love, support, motivation and advice, for dealing with my stress, and for “holding the fort” while I was away. I could not have done it without you.

This work was financially supported by the Research Foundation Flanders (FWO) under grant agreement number 1SB6219N, with additional funding from Vrije Universiteit Brussel.
# Contents

1 Introduction  
1.1 Introduction ........................................... 5  
1.2 Objectives and Contributions ................................... 8  
1.2.1 Main Contributions ........................................... 9  
1.2.2 Additional Contributions ...................................... 12  
1.3 Potential Impact ........................................... 13  
1.3.1 Intelligent Systems ........................................... 13  
1.3.2 Evolutionary Linguistics ...................................... 14  
1.3.3 Construction Grammar ........................................ 15  
1.4 Structure of the Dissertation ..................................... 16

2 Background and Technical Foundations  
2.1 Introduction ........................................... 20  
2.2 Language Games with Babel ..................................... 20  
2.2.1 Evolutionary Linguistics ...................................... 21  
2.2.2 The Language Game Paradigm ................................ 25  
2.2.3 Language Game Experiments ................................... 28  
2.2.4 Babel ..................................................... 30  
2.2.5 Meta-Layer Architecture ...................................... 32  
2.3 Computational Construction Grammar with FCG ................. 34  
2.3.1 Construction Grammar ......................................... 35  
2.3.2 Fluid Construction Grammar ................................... 38  
2.3.3 Language Processing as Problem Solving ....................... 39  
2.3.4 Building Blocks of FCG ........................................ 42  
2.3.5 Categorial Networks .......................................... 50  
2.3.6 Neural Heuristics ........................................... 54  
2.4 Procedural Semantics with IRL  
2.4.1 Procedural Semantics ......................................... 57  
2.4.2 Incremental Recruitment Language  
2.4.3 Building Blocks of IRL ........................................ 61
CONTENTS

6.3.3 Environment .................................................. 200
6.3.4 Interaction Script ............................................. 200
6.3.5 Intention Reading with IRL ................................. 204
6.3.6 Pattern Finding with FCG ................................. 209
6.3.7 Competition between Repair Strategies .......... 219
6.3.8 Alignment ......................................................... 222
6.3.9 Tutor Behaviour .............................................. 223
6.3.10 Learning Dynamics ......................................... 224
6.4 Experiments ...................................................... 225
6.4.1 Metrics .......................................................... 225
6.4.2 Main Results .................................................. 226
6.4.3 Probabilistic Tutor Strategy ............................... 233
6.5 Conclusion ......................................................... 235
6.5.1 Contributions .................................................. 236

7 Conclusions ...................................................... 241
7.1 Introduction ....................................................... 241
7.2 Achievements .................................................... 243
7.2.1 Large-Scale Computational Construction Grammar .... 243
7.2.2 Learning Grounded Concepts through Discrimination .... 244
7.2.3 Fully Explainable Grounded Language Processing System .... 245
7.2.4 Hybrid Procedural Semantics ................................. 245
7.2.5 Mechanistic Model of Intention Reading and Pattern Finding .... 246
7.3 Limitations and Avenues for Future Research .......... 247
7.3.1 Integration ...................................................... 248
7.3.2 Evaluating Explainability .................................... 248
7.3.3 Beyond Visual Concepts ..................................... 249
7.3.4 From Language Acquisition to Language Emergence .... 249
7.3.5 Symbolic/Sub-Symbolic Boundary .......................... 250
7.3.6 Scaling Intention Reading and Pattern Finding ........ 251
7.4 Final Remarks .................................................... 253

Appendix A Supplementary Materials for Chapter 4 ......... 289
Appendix B Supplementary Materials for Chapter 5 ......... 295
Appendix C List of Publications .................................. 299
Chapter 1

Introduction

1.1 Introduction

Language is the quintessential trait of human intelligence. Our linguistic systems are extremely well-suited for achieving our communicative goals, they are also immensely expressive, incredibly robust and they can be learned remarkably efficient and fast. These characteristics stem from the fact that linguistic systems are shaped by evolutionary processes (Smith and Szathmáry, 2000; Steels and Szathmáry, 2018). However, the exact mechanisms that regulate the variation, selection and self-organisation of linguistic structures, that drive the emergence and ongoing evolution of human languages and that bring to bear the properties exhibited by human languages are still heavily debated.

One way to unravel these mechanisms is to equip autonomous artificial agents with mechanistic models of them and simulate the acquisition, emergence and evolution of linguistic
communication systems. In fact, the development of these mechanistic models and agent-based communication systems offering human-like robustness, flexibility and adaptivity constitutes a fundamental challenge in artificial intelligence and is often claimed to be a necessary precondition for building truly intelligent systems (Mikolov et al., 2016; Van Eecke and Beuls, 2020). From a computational point of view, such systems are preferably transparent and explainable in human-interpretable terms as this facilitates the verification of their internal correctness and consistency (Lowe et al., 2019), and thereby elicits trust. Today, however, the communication systems of autonomous agents are still far removed from exhibiting these human-like properties as the design of adequate representations and learning mechanisms that lead to such systems is an unsolved problem.

Three main shortcomings can be identified in prior work on the acquisition of agent-based communication systems. At their core, these communication systems should be able to perform the basic function of language, namely to express meanings through utterances as a speaker, and to retrieve meanings underlying utterances as a listener. A first group of prior work is targeted towards learning communication systems from corpora of forms and meanings. While exploring interesting ideas, such approaches have only been demonstrated on inputs with limited complexity in terms of both forms and meanings, and often have access to additional information next to the raw inputs, such as a predefined lexicon or a priori segmented forms (Doumen et al., forthcoming), i.a. Dominey and Boucher (2005); Dominey (2005a,b, 2006); Chang (2008); Gaspers et al. (2011, 2016); Gaspers and Cimiano (2012, 2014) and Abend et al. (2017). In a second group of work, mappings between forms and meanings are learned via task-oriented, communicative interactions. While this approach is well suited for autonomous agents, it is always applied to specific linguistic phenomena (Doumen et al., forthcoming), such as Russian aspect (Gerasymova and Spranger, 2010, 2012), English spatial expressions (Spranger and Steels, 2015; Spranger, 2015, 2017), Hungarian agreement (Beuls et al., 2010), or word order in English noun phrases (Van Eecke, 2018, Ch. 7). A third group of prior work relies on the methodologies from the Multi-Agent Reinforcement Learning (MARL) paradigm and focusses on learning emergent communication protocols for solving a particular task (Van Eecke and Beuls, 2020), such as visual question answering (Das et al., 2017b), solving puzzles (Foerster et al., 2016), negotiation (Cao et al., 2018), reference (Lazaridou et al., 2016b), navigation (Sukhbaatar et al., 2016; Bogin et al., 2018; Mordatch and Abbeel, 2018), and coordination in self-driving cars (Resnick et al., 2018). While these approaches achieve impressive results on their respective tasks, they struggle to operationalise some of the key properties found in human communication systems, which has important repercussions on the emerged languages (Van Eecke and Beuls, 2020). Many of the aforementioned approaches rely on black-box architectures, with the MARL paradigm in particular being dominated by neural network-based methodologies. This makes it difficult to gain insights into the acquired languages and, more importantly, the underlying processes and mechanisms that led to them.

This dissertation aims to push forward the state of the art in this area of research by pre-
senting novel representations and learning mechanisms that bring to bear bidirectional communication systems for autonomous artificial agents, that exhibit the key desirable properties of human languages, such as expressiveness, robustness, efficiency and adaptivity, that operate on a larger scale compared to previous work both in terms of the complexity of the input and the linguistic phenomena considered, and that focus on computational representations and processes that are transparent and human-interpretable.

The methodology for modelling the communication systems of artificial agents adopted in this dissertation is the language game paradigm (Steels, 1995). This paradigm tackles the question of how linguistic conventions can emerge through local interactions and coordination in a population of autonomous agents that are situated in their native environment. To investigate the many aspects that are involved in such a modelling effort, the language game paradigm requires a whole-systems approach (Steels et al., 2012). In particular, setting up a language game necessitates the operationalisation of mechanisms taking place on the sensorimotor, conceptual and linguistic level, all working together to implement the basic function of language. This requires theories, methodologies, tools and techniques that are well-suited for this task, sufficiently advanced and can be integrated with each other, especially for dealing with the emergence and evolution of more complex phenomena such as grammatical structures. One theory from linguistics that has been particularly successful in capturing the basic function of language is construction grammar (i.a. Goldberg (1995); Kay and Fillmore (1999); Croft (2001)). In this dissertation, I therefore rely on Fluid Construction Grammar (FCG) (Steels, 2011a, 2017; van Trijp et al., 2022, https://fcg-net.org), the most advanced computational construction grammar formalism. Next to this, procedural cognitive semantics (Woods, 1968; Winograd, 1972; Johnson-Laird, 1977) provides a framework that elegantly integrates the meaning of natural language utterances in sensorimotor processing. Therefore, I use Incremental Recruitment Language (IRL) (Van den Broeck, 2008; Spranger et al., 2012b), a special-purpose formalism for representing and processing procedural semantic representations. Both of these tools are integrated in the Babel software package for running multi-agent language game experiments (Steels and Loetzsch, 2010, https://emergent-languages.org).

The contributions of this dissertation focus on the representation, processing, and learning of linguistic structures that are on the one hand situated on the conceptual level, and on the other hand on the morpho-syntactic and semantic level. Conceptual structures allow autonomous agents to link the experiences in their low-level, sensorimotor data streams to higher-level, symbolic concepts that are meaningful in their environment and for the task at hand, and with which they can reason. Learning these structures requires to identify and extract relevant patterns in the agents’ sensorimotor data and update them as more observations become available. Morpho-syntactic and semantic structures further allow the agents to express those concepts through utterances that convey information on how these concepts interact in a particular situation and dampen referential ambiguities for their interlocutors. Learning these structures is extremely difficult as only the morpho-
syntax can ever be observed in communicative interactions and the underlying semantics thus needs to be reconstructed. The methodology for learning these morpho-syntactic and semantic structures constitutes the most important contribution of this dissertation.

The remainder of this chapter consists of the following sections. In Section 1.2, I lay out the objectives of this dissertation and the concrete contributions that tackle them. Afterwards, I discuss the potential impact of these contributions in Section 1.3. Finally, Section 1.4 provides an overview of the structure of this dissertation.

1.2 Objectives and Contributions

The primary objective (O1) of this dissertation is the introduction of novel representations and learning mechanisms that enable autonomous agents to acquire linguistic structures suitable for solving communicative tasks in their environment and that bring to bear the key desirable properties of human languages, such as robustness, flexibility, adaptivity, learning efficiency and expressiveness. These representations and learning mechanism either extend earlier work within the language game paradigm, or they are developed from scratch and inspired on theories, empirical findings and methodologies from artificial intelligence, linguistics and cognitive science. The ultimate goal of this objective is to advance the state of the art in human-like communication systems for autonomous agents, which in turn can be beneficial in two major areas. On the one hand, these developments allow for conducting more advanced experiments on the emergence and evolution of language and thereby provide valuable insights into human languages and cognition. On the other hand, the advancements facilitate the design of the next wave of intelligent systems where agents and humans communicate through natural language in a way that is more robust, natural, coherent, and explainable. Examples of such systems include human-robot interaction systems, conversational agents and intelligent tutoring systems.

A secondary objective (O2) is to validate the novel representations and learning mechanisms through case studies that tackle challenging communicative tasks in concrete environments. In this dissertation, I focus on the task of visual question answering (VQA). First introduced by Antol et al. (2015), the VQA task involves answering natural language questions about images, which requires the design and integration of perceptual, linguistic and reasoning abilities in artificial systems. I focus particularly on the CLEVR benchmark task designed by Johnson et al. (2017a). This benchmark was specifically designed to diagnose the linguistic and reasoning capabilities of VQA systems and focus less on the perceptual abilities. In particular, the benchmark provides computationally rendered images of 3D objects that are relatively easy to process using off-the-shelf techniques, paired with natural language questions that feature a wide range of linguistic structures, such as noun phrases, prepositional phrases, anaphora, conjunction and subordination, and test a wide variety of reasoning skills, such as counting, attribute identification, logical operations,
spatial operations and comparison. This level of complexity, both in terms of the provided input and the variation of linguistic phenomena, combined with the focus on linguistic and reasoning abilities, make the CLEVR benchmark task well suited for the purposes of this dissertation.

1.2.1 Main Contributions

These two objectives are tackled by five main contributions. The first contribution, called the CLEVR grammar, constitutes a baseline solution to the CLEVR benchmark task. It captures all the conceptual, morpho-syntactic and semantic structures required to map all questions of the CLEVR dataset to their underlying meaning representations (and vice versa), and use these meaning representations to compute the answers. The other contributions consist of new representations and learning mechanisms that are applied for learning the conceptual, morpho-syntactic and semantic structures of the CLEVR grammar. Specifically, the second, third and fourth contribution focus on learning conceptual structures, while the fifth contribution focusses on learning morpho-syntactic and semantic structures. The acquired linguistic structures are evaluated through the communicative task of visual question answering task on the CLEVR dataset. In what follows, I describe these contributions in more detail.

The CLEVR grammar (C1)

Two kinds of representations are central in constructing the CLEVR grammar. On the one hand, all linguistic knowledge necessary for the benchmark is captured by an inventory of form-meaning mappings, or constructions, formalised using FCG. These constructions offer complete coverage of all questions of the CLEVR dataset. In other words, the grammar is able to map between all questions and their underlying meaning representations. These meanings are represented in terms of procedural cognitive semantics, implemented in IRL. In procedural cognitive semantics, meaning is captured in the form of constraint networks that can be executed algorithmically. When executed over a symbolic annotation of the CLEVR images, the IRL meaning representations correctly compute the answers to all questions of the dataset. The CLEVR grammar, including both the FCG constructions and the IRL meaning representations, thus effectively solves the CLEVR benchmark task on the symbolic level.

I show that the CLEVR grammar not only allows the CLEVR benchmark to be solved in a transparent and human-interpretable manner, it is also easily extensible, more expressive than the CLEVR dataset, does not require any annotated training data and operates both in comprehension, i.e. mapping questions to meanings, and production, i.e. mapping meanings to questions. Furthermore, the CLEVR grammar constitutes one of the first truly large-scale FCG grammars, covering more than one million utterances in both directions of processing. It thereby contributes to the scaling of computational construction gram-
marts and demonstrates the potential of recent advances in Fluid Construction Grammar, such as the introduction of a new high-level notation by Van Eecke (2018).

Methodology for learning grounded concepts through discrimination (C2)

I present an interactive learning approach, through the language game paradigm, where an agent learns to extract meaningful symbolic concepts from continuous sensorimotor observations of its environment in order to solve an object reference task. The concept representation is inspired by prototype theory (Rosch, 1973) and the learning mechanisms extend earlier experiments within the language game paradigm (Wellens, 2012). The notion of discrimination, i.e. maximally separating one object with respect to others in terms of perceptual observations, plays a central role in shaping the concepts. The novel methodology is applied to the CLEVR dataset for learning the various perceptual concepts it contains, such as colours, shapes and materials. Through a number of experiments, I show that the proposed methodology results in a repertoire of concepts that can be learned efficiently and rapidly, generalises well to unseen instances, is adaptive to changes in the environment and offers transparent and human-interpretable insights into the agent’s processing and memory. These properties are highly desirable for robotics and interactive task learning, and make the proposed methodology readily applicable to tackle various tasks in those domains. Additionally, the experiments contribute to the research on the emergence and evolution of conceptual systems within the language game paradigm.

End-to-end human-interpretable grounded language processing system (C3)

I present the integration of the repertoire of grounded concepts, acquired through the methodology of C2, in higher-level reasoning tasks. In particular, the acquired perceptual concepts are integrated with the procedural semantic representations of the CLEVR grammar in order to tackle the CLEVR benchmark task. This integration constitutes a grounded language processing system that is end-to-end human interpretable, ranging from the constructional language processing (C1) to the representation and acquisition of concepts (C2) and their integration in the symbolic processing of the procedural semantic representations (C1). Within the domain of visual question answering, that is traditionally dominated by block-box architectures, this constitutes a radically different approach that focusses on tackling the VQA task in a way that is transparent, human-interpretable, adaptive and open-ended. While I demonstrate the fully explainable grounded language processing system in a VQA setting, the same methodology can be followed to operationalise similar systems that require a combination of perceptual, linguistic and reasoning abilities.
1.2. OBJECTIVES AND CONTRIBUTIONS

Methodology for learning grounded concepts through hybrid procedural semantics (C4)

I contrast the methodologies of C2 and C3 with a novel methodology named hybrid procedural semantics. Hybrid procedural semantics immediately integrates the grounded concept learning task within the visual question answering task. In particular, the grounded concepts are captured by small, modular and highly specialised image-processing neural networks. These are subsequently integrated in the procedural semantic representations of the CLEVR grammar, interleaved with existing, symbolic functionality. I show that the hybrid procedural semantics methodology offers an elegant and flexible way to combine symbolic and sub-symbolic techniques and achieves results on the CLEVR benchmark task that are competitive with the state of the art. Furthermore, I show that the neural networks responsible for capturing concepts can be learned efficiently and are designed to enable the transparency and open-endedness of the hybrid procedural semantics methodology. The main novelty of this methodology is thus the modular and flexible integration of highly accurate sub-symbolic techniques for pattern recognition with symbolic reasoning processes through procedural semantics. Next to VQA, hybrid procedural semantics is directly applicable in a wide variety of other tasks that require natural language interaction with and reasoning over images, knowledge graphs or multiple such sources of information.

Mechanistic model of intention reading and pattern finding for learning grammar (C5)

For the largest experiment and most important contribution of this dissertation, I turn to the acquisition of morpho-syntactic and semantic structures. Specifically, in the previous contributions, I focussed on learning the perceptual concepts present in the CLEVR dataset and integrated them in the procedural semantic representations of the CLEVR grammar in order to solve the CLEVR benchmark task. I thereby relied on the hand-crafted FCG constructions to perform the linguistic analysis of the questions. The aim of this contribution is to learn the FCG constructions of the CLEVR grammar through task-oriented, communicative interactions. This requires learning a set of form-meaning mappings that cover the observed questions of the CLEVR dataset, but also novel ones in the same style. Crucially, only the questions themselves are ever observed in communicative interactions, implying that the agent needs to reconstruct their underlying meanings. However, the space of possible meanings underlying a previously unobserved utterance is infinitely large. Managing the complexity of this space and intelligently navigating it constitutes the main challenge of this experiment.

I overcome this challenge by relying on two cognitive processes that play a central role in usage-based theories of language acquisition, namely intention reading and pattern finding (Tomasello, 2003, 2009b). Specifically, I introduce a mechanistic model of intention reading and integrate it with an existing mechanistic model of pattern finding. The latter was
designed to learn form-meaning mappings from annotated corpora (Doumen et al., forthcoming). Intention reading enables the agent to hypothesise about the intended meaning of an observed utterance, while pattern finding enables the agent to construct abstract schemata that capture generalisations over forms and meanings. I argue that the interplay of these cognitive capacities reduces the search space of possible meanings and ultimately allows the agent to incrementally acquire an open-ended FCG grammar that is adequate for solving the communicative task, in this case question answering, through situated and local interactions.

I show that the cognitive capacities of intention reading and pattern finding are operationalised through a number of human-interpretable learning operators that allow the grammar to be acquired incrementally, efficiently and rapidly. These learning operators remain ever-adaptive and result in a grammar that is transparent, open-ended and facilitates both language comprehension and production.

In sum, this methodology is a major contribution to the research on the emergence of grammar within the language game paradigm and it advances the state of the art with respect to the development of human-like communication systems for autonomous agents. Similar to Doumen et al. (forthcoming), the presented experiment provides computational evidence for the cognitive plausibility of theories from usage-based language acquisition, in particular intention reading and pattern finding, and corroborates the theoretical underpinnings of the field of construction grammar. In contrast to Doumen et al. (forthcoming), the agent does not rely on annotated corpora of forms and meanings, but faces the challenging task of reconstructing meanings through intention reading over the course of a series of communicative interactions.

1.2.2 Additional Contributions

Apart from the five main contributions that are directly aimed at tackling the two objectives of this dissertation, I describe here one additional contribution that was developed during my PhD project.

Neural heuristics for computational construction grammar (C6)

The field of construction grammar aims to develop an all-encompassing scientific theory of language. In computational constructional grammar, these theoretical insights and analyses are formalised into concrete processing models (Van Eecke and Beuls, 2018). FCG, in particular, handles constructional language processing as a state-space search problem (Bleys et al., 2011; Van Eecke and Beuls, 2017), looking for the sequence of constructions that successfully maps an utterance onto its meaning representation or vice versa. A major issue in FCG and computational construction grammar at large is scaling, as the modelling and processing of a large number of constructions and all of their intricate relations soon becomes intractable (van Trijp et al., 2022).
1.3. POTENTIAL IMPACT

Inspired by recent successes in artificial intelligence that intelligently combine neural networks with traditional search techniques (e.g., AlphaGo (Silver et al., 2016)), recurrent neural networks are integrated into the FCG search process. Specifically, encoder-decoder neural networks are trained on the input of the comprehension or production processes together with the sequences of constructions that successfully analyse those inputs. The trained networks can then be used to predict heuristic values during the FCG search process to guide the exploration of branches and the application of particular constructions. A case study on the CLEVR grammar shows that the neural heuristic methodology outperforms other state-of-the-art approaches that are aimed at reducing the search space in FCG. This case study has been submitted as Van Eecke et al. (subm). The neural heuristics methodology primarily contributes to the field of computational construction grammar in that it helps to overcome the issue of intractability, which has an impact both theoretically and practically (Van Eecke et al., subm). On the theoretical level, being able to efficiently process a large number of constructions allows construction grammarians to gain more insight into the intricate relations of constructions and how a large number of them cooperates in analysing utterances. On the practical level, the advances obtained by scaling computational construction grammar can in turn lead to breakthroughs in the automatic learning of construction grammars (e.g., Doumen et al. (forthcoming) and C5), which has a broader impact in usage-based linguistics research (Diessel, 2015), language acquisition models (Tomasello, 2003) and language technology applications (Nevens et al., 2019a; Willaert et al., 2020, 2021, 2022; Beuls et al., 2021).

1.3 Potential Impact

The contributions presented in this dissertation have a potential impact in at least three areas of research.

1.3.1 Intelligent Systems

Ever since Winograd (1972)’s SHRDLU, researchers in artificial intelligence have been developing systems in which intelligent agents need to communicate, either with humans or with each other, through natural language. Nowadays, such systems are omnipresent and tackle a wide variety of tasks. Examples include visual question answering systems, conversational agents, navigation systems, personal assistants, human-robot interaction systems and intelligent tutoring systems. The key to success in such systems is three-fold. First, it is crucial to develop a precise understanding of the user’s natural language query. This requires powerful natural language processing capabilities that can analyse utterances in order to produce concrete meaning representations. Second, the meaning representations should facilitate reasoning within the domain of the application in order to fulfil the user’s query. Third, intelligently conversing with users requires these processes to be bidirectional, such that concrete meaning representations grounded in the domain
CHAPTER 1. INTRODUCTION

of the application can also be conceptualised and formulated through natural language utterances.

The contributions presented in this dissertation have the potential of being incorporated in a wide range of intelligent systems of the kind just described. In particular, the presented representations and learning mechanisms focus both on conceptual structures, bridging the gap from low-level, sensorimotor input to a higher level where reasoning and language capabilities reside, and morpho-syntactic and semantic structures, i.e. grammar. In other words, the learning mechanisms focus on constructing meaning representations that are grounded in the agents’ environment. Moreover, the properties of robustness, flexibility, adaptivity and open-endedness are central. These properties ensure that there is no separate training phase, but the intelligent systems can keep on learning and adapting indefinitely even when the task or the environment changes. Furthermore, the presented methodologies focus on transparency and interpretability, which allows users to inspect the internal reasoning and decision making processes of the intelligent systems. Finally, the presented methodologies support bidirectional language processing, which allows the intelligent systems, together with the necessary reasoning components, to produce answers, ask for clarifications, and fill potential knowledge gaps in a more natural and coherent manner.

In terms of the visual question answering task in particular, this dissertation presents radically different approaches to tackle this task in a field that is dominated by neural network-based approaches. These contributions achieve results that are competitive with the state of the art whilst possessing the aforementioned highly desirable properties.

As a concrete example of the potential impact of this dissertation in terms of intelligent systems, the hybrid procedural semantics methodology (C4) is currently being extended for use in conversational agents, and in particular to tackle the task of visual dialogue. In this task, the procedural semantic representations not only elegantly and flexibly combine perceptual and reasoning abilities, they additionally include a symbolic memory component that keeps track of what has been said in the conversation. This further highlights the benefits of the hybrid approach since sub-symbolic perceptual elements can be seamlessly integrated both in symbolic reasoning and in symbolic memory, but also symbolic memory elements can be as easily retrieved and related to sub-symbolic perceptual elements. When successful, further extensions that can additionally incorporate large amounts of background world knowledge by including knowledge graphs or ontologies come within reach.

1.3.2 Evolutionary Linguistics

The potential impact of my dissertation in the field of evolutionary linguistics resides in the advancement of the research on the acquisition of conceptual systems (C2) and the acquisition of grammar (C5). By building further on these contributions, many exciting new
1.3. POTENTIAL IMPACT

experiments come within reach. First of all, there is the integration of the aforementioned contributions. Specifically, the acquisition of grounded concepts could be supplemented with a system that abstracts meaningful semantic categories over these concepts and in turn uses those semantic categories to guide the intention reading and pattern finding processes during grammar learning. Such an experiment would allow an intelligent agent that starts out from scratch to progress through several learning stages and end up with a bidirectional grammar, e.g. for asking and answering questions, where the perceptual concepts that are part of the grammar are grounded in the agent’s sensorimotor experience. Such an experiment would shed light on a large number of mechanisms that are required for the acquisition of conceptual, morpho-syntactic and semantic structures, and particularly how these mechanisms interact. Second, there is the question of how these structures, both conceptual and grammatical, originate and how a community of language users achieves a consensus over them. This requires multi-agent experiments where all agents start without any linguistic structures, but are equipped with precise mechanistic models of the invention, adoption and alignment mechanisms that are required for establishing both the conceptual and grammatical structures from the ground up. The results of such experiments will lead to a better understanding of the mechanisms involved in the emergence and evolution of conceptual, morpho-syntactic and semantic structures, and contribute to the hypothesis that linguistic structures emerge through communicative interactions using general cognitive capacities, instead of being innate.

1.3.3 Construction Grammar

The contributions presented in this dissertation have a potential impact on both construction grammar theory and computational construction grammar. In terms of construction grammar theory, both the CLEVR grammar (C1) and the presented methodology for learning construction grammars through intention reading and pattern finding (C6) corroborate many of the theoretical findings of the field, such as the free combination of constructions (Goldberg, 2006, p. 22), statistical pre-emption (Goldberg, 2011), and the co-emergence of constructions and construction-specific, functionally motivated grammatical categories (Croft, 2001). From a computational point of view, the CLEVR grammar (C1) is one of the first computational construction grammars to operate on this scale in both directions of processing in a computationally efficient manner. This underlines the potential of Fluid Construction Grammar and its recent additions, such as the new high level notation for formalising constructions or the ability to hash constructions for more efficient retrieval (Van Eecke, 2018). I further contribute to the scaling of computational construction grammar by presenting a concrete methodology for learning heuristics that substantially optimise the search process involved in constructional language processing (C6). Both of these contributions (C1 and C6) constitute concrete technical advances in computational construction grammar, and in particular in Fluid Construction Grammar. These advances have the potential to push forward the field of computational construction grammar, par-
particularly by overcoming scaling issues, which in turn allows constructions grammarians to computationally verify their theories on a larger scale and facilitates the use of computational construction grammar in usage-based linguistics research, corpus studies, historical linguistics, language acquisition models and language technology applications (van Trijp et al., 2022).

1.4 Structure of the Dissertation

The remainder of this dissertation is structured as follows:

- **Chapter 2: Background and Technical Foundations.** In this chapter, I lay out the technical foundations for operationalising the contributions that are presented in subsequent chapters, together with the broader research context in which these are embedded. Concretely, I introduce the Babel software package, Fluid Construction Grammar (FCG), and Incremental Recruitment Language (IRL). The neural heuristics methodology (C6) is discussed in the context of Fluid Construction Grammar.

- **Chapter 3: Data.** This chapter introduces the CLEVR benchmark dataset (Johnson et al., 2017a) for the task of visual question answering (VQA). This benchmark task, or derivatives thereof, will be used to validate the novel representations and learning mechanisms introduced in subsequent chapters. Afterwards, this chapter presents the CLEVR grammar (C1). I motivate why the CLEVR grammar was made in light of this dissertation and why FCG and IRL are good candidates for operationalising this and other VQA systems.

- **Chapter 4: Learning Concepts through Discrimination.** This chapter introduces the methodology for learning grounded concepts from continuous sensorimotor observations through situated communicative interactions (C2). I first discuss background material on prototype theory and related work on grounded concept learning. Afterwards, I introduce the methodology and present the various experiments that are designed to highlight the methodology’s desirable properties. This chapter also includes the integration of the acquired concepts with the procedural semantic representations of the CLEVR grammar (C3). Experimental results of this integration are presented and discussed.

- **Chapter 5: Learning Concepts as Neural Modules.** This chapter presents the hybrid procedural semantics methodology (C4). After discussing related approaches that tackle VQA tasks, I introduce the methodology itself and how it was applied to the CLEVR dataset. This includes the design and training of the neural networks and their integration in IRL. Afterwards, I present the evaluation results of this integration on the CLEVR benchmark task and compare them to the state-of-the-art approaches discussed at the beginning of the chapter.
• **Chapter 6: Learning Morpho-Syntactic and Semantic Structures.** This chapter presents the mechanistic model of intention reading and its integration with that of pattern finding (C5). I situate this model both with respect to theories of usage-based language acquisition and computational models of grammar learning. Afterwards, I introduce the language game that is set up for learning the CLEVR grammar and discuss the learning operators that implement intention reading and pattern finding. These learning operators fully exploit the capabilities of both FCG and IRL. Finally, I present the experimental results of the language game experiment and discuss their theoretical and practical implications.

• **Chapter 7: Conclusion.** In the final chapter, I summarize the contributions of this dissertation, discuss its achievements and elaborate on future directions of research.
Chapter 2

Background and Technical Foundations

2.1 Introduction ......................................................... 20
2.2 Language Games with Babel .................................. 20
  2.2.1 Evolutionary Linguistics ................................. 21
  2.2.2 The Language Game Paradigm ......................... 25
  2.2.3 Language Game Experiments ......................... 28
  2.2.4 Babel ...................................................... 30
  2.2.5 Meta-Layer Architecture ............................... 32
2.3 Computational Construction Grammar with FCG ........... 34
  2.3.1 Construction Grammar ................................. 35
  2.3.2 Fluid Construction Grammar ......................... 38
  2.3.3 Language Processing as Problem Solving ............. 39
  2.3.4 Building Blocks of FCG ............................... 42
  2.3.5 Categorial Networks ................................. 50
  2.3.6 Neural Heuristics ....................................... 54
2.4 Procedural Semantics with IRL ............................... 57
  2.4.1 Procedural Semantics ................................. 57
  2.4.2 Incremental Recruitment Language .................... 59
  2.4.3 Building Blocks of IRL .............................. 61
  2.4.4 Evaluating Semantic Networks ........................ 64
  2.4.5 Composing Semantic Networks ....................... 69
  2.4.6 Chunking .................................................. 74
  2.4.7 Flexible Interpretation ............................ 76
Preface


2.1 Introduction

In this chapter, I introduce the tools and techniques that are used to operationalise subsequent chapters of this dissertation together with the broader research context in which these technical foundations are embedded. Concretely, in Section 2.2, I introduce the research field of evolutionary linguistics and situate the language game paradigm within this field. Afterwards, I discuss this paradigm in greater depth and provide an overview of Babel, a flexible toolkit for implementing and running multi-agent language game experiments on the emergence of language. Next, in Section 2.3, I introduce the field of construction grammar followed by an introduction to Fluid Construction Grammar (FCG), the most advanced computational construction grammar formalism. I rely on FCG to operationalise the language processing capabilities of autonomous agents in the language game experiments. Finally, Section 2.4 introduces procedural cognitive semantics and the Incremental Recruitment Language (IRL) system. IRL is a computational formalism for representing, evaluating and automatically constructing procedural semantic representations, and facilitates the integration of the meaning of utterances in the sensorimotor experiences of autonomous agents. The aforementioned sections include contributions that I made during my PhD project to each of these tools, namely Babel, FCG and IRL. A summary of this chapter and an overview of these contributions is presented in Section 2.5. Finally, I note that related work describing the state of the art and relevant background literature for each of the experiments will be provided separately in the following chapters.

2.2 Language Games with Babel

In this section, I first introduce the field of evolutionary linguistics, which studies the origins and evolution of natural languages (Section 2.2.1). This topic can be studied from three different perspectives, as identified by Steels (2012b, p. 1-2): the biological perspective, the social perspective and the cultural perspective. After briefly describing the goals and
methodologies of these perspectives, I zoom in on the cultural perspective and introduce three experimental paradigms that allow to study the emergence of linguistic conventions. The contributions of this dissertation are situated within one of these paradigms, namely the language game paradigm. After discussing the language game paradigm more elaborately in Section 2.2.2, I provide an overview of prior work that fits within this paradigm (Section 2.2.3) and I introduce the Babel software package that is specifically designed to operationalise language game experiments, including all experiments in this dissertation. Finally, I discuss Babel’s meta-layer architecture (Section 2.2.5) that enables agents to flexibly handle problem solving and learning.

2.2.1 Evolutionary Linguistics

Researchers in evolutionary linguistics argue that languages are dynamic systems and that linguistic structures emergence through communicative interactions that are influenced by the context, the interlocutors and past experiences (Hopper, 1987; Jaspersen et al., 1994) (Van Eecke, 2018, p. 16). Underlying this are evolutionary processes, such as variation and selection, taking place within the linguistic system (Smith and Szathmáry, 2000; Steels and Szathmáry, 2018) and shaping language according to the communicative goals and environmental conditions faced by the language community (Spranger, 2016, p. 2). This view on the evolution of natural languages is in contrast to the view that linguistic structures are innate and form a stable, universal grammar that underlies all natural languages (Chomsky, 1986).

Three Perspectives on Language Evolution

Evolutionary linguistics is a highly interdisciplinary field of research that brings together evidence from a wide spectrum of scientific disciplines. Concretely, the evolution of natural languages can be studied from three different perspectives (Steels, 2011b) (Steels, 2012b, p. 1-2) (Spranger, 2016, p. 4-6) (Van Eecke, 2018, p. 16-17):

- **Biological.** From a biological perspective, the main questions underlying language evolution concern the neuro-biological structures and processes that are necessary for language, where these are encoded in the human genome, when these have evolved, what their genetic bases are and whether any precursors in non-human primates or other species can be found. From this perspective, one can investigate how these neuro-biological structures and processes influence the general capacity for language, or how they influence particular languages. These questions are studied within the Darwinian genetic evolution framework used in evolutionary biology. Among others, this includes work by Dediu (2007); Bickerton and Szathmáry (2009); Fitch (2010); de Boer (2012) and Arbib (2012).

- **Social.** This perspective investigates the social mechanisms that are prerequisites for language emergence, or have influenced the evolution of language. The former
include proposals such as theory of mind (Dunbar, 1998), joint attention (Tomasello, 1995; Carpenter et al., 1998), imitation learning (Tomasello, 1992) or shared intentionality (Tomasello et al., 2005). Examples of the latter include (changes in) the size of communities, the social relations within communities, the need to cooperate on complex tasks and the role of trust (see e.g. Knight et al. (2000), Tomasello (2003) and Dor et al. (2014)). Methodologies from the fields of anthropology and social science are used to study this perspective on language evolution.

• **Cultural.** Languages evolve as a consequence of their use in communication, making them a cultural phenomenon. Cultural evolution can affect any unit of language, including speech sounds, concepts, words, semantic structures, morphological structures, syntactic structures or discourse structures, and encompasses the appearance, propagation, change, erosion and disappearance of these units. The cultural perspective of language evolution deals with the mechanisms underlying these evolutions and the principles that allow languages to become conventionalised. If these underlying mechanisms can be unravelled and one assumes that these mechanisms have been operating since the very first languages, their origins can in principle be reconstructed (Heine and Kuteva, 2007). To unravel these mechanisms, methodologies and processes from linguistics, such as grammaticalization (Hopper and Traugott, 2003), evolutionary biology, such as variation, selection, self-organisation and emergent functionality (Steels, 2012b), and artificial intelligence, such as multi-agent systems (Steels, 1999), evolutionary computation (Smith et al., 2003; Kirby et al., 2008), or reinforcement learning (Foerster et al., 2016; Lazaridou et al., 2016b) are used.

Despite that any general theory of language should cover all three of these perspectives (Steels, 2012b, p. 1), they can also be studied independently from each other. This is because language evolution processes at the biological, social and cultural level take place at different time-scales. There are, however, strong interactions between these perspectives, as illustrated by Steels (2012b, p. 3) and Van Eecke (2018, p. 17). For example, theories of cultural evolution cannot rely on unrealistic assumptions from the social perspective, e.g. central control over language, or the biological perspective, e.g. mind-reading capabilities. Moreover, progress from one perspective has repercussions in the other perspectives, resulting in an upward spiralling process. For instance, increased social capabilities require increased linguistic capabilities, which in turn increases the required brain capacity, which then further allows for increased social complexities and so on.

In this dissertation, I focus on language evolution from the cultural perspective. Theories from this perspective should not only be able to explain the emergence of specific linguistic phenomena, but also provide a general explanation of how and why languages emerge or change through cultural transmission (Steels, 2012b, p. 3).
## Experimental Paradigms

Within the cultural perspective on language evolution, there are three commonly used experimental paradigms (van Trijp, 2008, p. 39-42) (van Trijp, 2016, p. 11-12) (Van Eecke, 2018, p. 18-20) (Van Eecke and Beuls, 2020):

**Iterated Learning** investigates how the structure of languages are influenced by innate learning biases in the language learners when language is transmitted from one generation to the next (i.a. Kirby and Hurford (2002); Smith et al. (2003); Kirby et al. (2004); Brighton et al. (2005) and Kirby et al. (2008)). A central idea within the iterated learning paradigm is that structural language features arise because languages need to pass through the *transmission bottleneck*. Specifically, this bottleneck refers to the fact that it would be infeasible to pass over the infinitely large set of all possible utterances across generations. Therefore, language learners (children) only observe a small amount of learning data from which they need to reconstruct the language of their teachers (parents). This results in a pressure to obtain linguistic competence that leads the learners to overgeneralise based on their innate learning biases. In turn, these overgeneralisations propagate to the next generation when the learners become teachers themselves. This process continues over many generations until a stable point is reached. Experiments have shown that this process indeed allows systematic and compositional languages to emerge.

The iterated learning paradigm has been operationalised through four different methodologies. First, agent-based models have been used to investigate the learner’s innate biases that could steer the language toward compositional and recursive syntax (Kirby, 2001, 2002; Brighton et al., 2005) and towards one-to-one mappings between meanings and forms (Smith, 2002, 2004). Second, mathematical models have been used to develop convergence proofs (Griffiths and Kalish, 2007). Third, experiments with human participants have been used to validate the outcomes of agent-based simulations and thereby reveal the innate learning biases in humans (Kirby et al., 2008). Finally, the iterated learning paradigm has been integrated in artificial agents that are equipped with deep neural networks (i.a. Cogswell et al. (2019); Ren et al. (2020)). The aim of these approaches is to improve systematicity and compositionality in emergent communication scenarios within the Multi-Agent Reinforcement Learning (MARL) paradigm (see below).

In sum, the experiments conducted through the iterated learning paradigm convincingly show that structural language features can emerge through generational transmission, the transmission bottleneck that this entails and innate learning biases. However, these experiments do not focus on achieving communicative success between agents within a single generation, and the lack of communicative success does not affect the linguistic behaviour of the agents. Instead of interactively sharing, coordinating and aligning linguistic structures, there is a passive transmission of language from one generation to the next.
Multi-Agent Reinforcement Learning (MARL), on a high level, deals with a population of agents that are situated in an environment and learn to perform certain actions in certain states (called a policy) as to maximise their individual future rewards. This paradigm seems well-suited to model the conditions under which human languages emerge and evolve given that the high-level description of MARL can be easily translated to an ‘emergent communication’ scenario. In MARL, environments may be partially observable, requiring the agents to set up a communication protocol in order to complete a task. A speaker agent can then learn a policy to utter a particular utterance (action) in order to achieve the communicative goal (state). Conversely, the policy of a listener agent corresponds to performing an action in the environment that is relevant for the communicative goal, e.g. pointing to an object in a reference task, when observing a particular utterance (state). The reward of both speaker and listener reflects the communicative success of an interaction, e.g. positive reward when successful and negative reward otherwise.

Emergent communication research within the MARL community has been applied to a wide range of tasks that typically have a tight connection to real-world applications (Van Eecke and Beuls, 2020), such as visual dialog (Das et al., 2017b), solving puzzles (Foerster et al., 2016), negotiation (Cao et al., 2018), navigation (Sukhbaatar et al., 2016; Bogin et al., 2018; Mordatch and Abbeel, 2018; Mul et al., 2019), and coordination in self-driving cars (Resnick et al., 2018). However, the bulk of research is focussed on reference and discrimination tasks (i.a. Lazaridou et al. (2016a,b, 2018); Bouchacourt and Baroni (2019); Graesser et al. (2019); Chaabouni et al. (2021b,a)). The main goal of all of these experiments is learning a communication protocol that is effective for solving the given task. However, the conditions faced by the agents in these tasks are often far-removed from those under which human languages have emerged and continuously evolve (Lowe et al., 2019). For instance, many experiments are limited to only speaker agents or only listener agents, with possibly different internal architectures (e.g. Das et al. (2017b); Bogin et al. (2018); Lazaridou et al. (2016b,a, 2018); Chaabouni et al. (2021a)). This makes it impossible for these agents to switch from the speaker role to the listener role or vice versa. When bidirectional communication is considered, comprehension and production processes are often not integrated with each other, allowing for separate languages to be learned for speaking and listening (e.g. Bouchacourt and Baroni (2019); Graesser et al. (2019)). MARL experiments are often limited to two agents (e.g. Lazaridou et al. (2016a, 2018); Das et al. (2017b)), which circumvents many of the problems of learning linguistic conventions, since every agent participates in every interaction. When a multi-agent setting is adopted, learning is not necessarily decentralised (e.g. Foerster et al. (2016)), which would indicate central control over language. Additionally, agents are not always fully autonomous, but possess mind-reading or broadcasting capabilities (e.g. Mordatch and Abbeel (2018)). By the lack of local interactions between autonomous agents with decentralised learning, the acquired communication systems cannot benefit from the robustness that is brought forward by self-organising systems. Finally, the linguistic inventories of the agents are often prede-
Language Games study the emergence of linguistic conventions in a population of artificial agents through routinised local interactions and coordination (Steels, 1995, 1997a, 2012b). The experiments presented in this dissertation are operationalised through the language game paradigm. Having situated this paradigm within the cultural perspective on language evolution, I will now use the remainder of Section 2.2 to describe this paradigm in greater detail.

2.2.2 The Language Game Paradigm

The language game paradigm tackles the question of how an effective and efficient communication system can emerge in a population of agents through a series of situated, communicative interactions. Each game is played between two agents from the population that are randomly assigned the discourse roles of speaker and listener. These agents follow a particular turn-taking routine or interaction script, where each turn may involve symbolic communication, physical actions or gesturing. Crucially, all agents are situated in an environment that may be simulated or physical, they are fully autonomous in that there is no central control, mind-reading or broadcasting, and their interactions are local and decentralised. A language game experiment models a particular communicative task that necessitates the acquisition or emergence of a particular natural language-like phenomenon and its conventionalisation within the population of agents. If successful, the population of agents will have developed a shared communication system that is adequate for the communicative task in their native environment.

Complex Adaptive Systems

In the language game paradigm, a community of language users can be seen as a complex adaptive system (Steels, 2000b). Language game experiments thereby build on four main principles:

- **Variation and selection**, for generating variants of linguistic structures and selecting them based on fitness under a number of selective pressures faced by a language...
community (Steels, 2012b, p. 14-15), such as communicative success (Clark and Brennan, 1991), cognitive effort (Fitch, 2000; de Boer, 2000) and social conformity (Nettle, 1999)

- **Self-organisation**, for organising a population-wide coherent language through local interactions and alignment (Garrod and Doherty, 1994; Pickering and Garrod, 2006) (Steels, 2012b, p. 22-26)

- **Level formation**, by (possibly hierarchical) schema formation over semantic and syntactic structures (Steels, 1997b)

- **Reinforcement learning**, by creating a positive feedback loop between success and use of linguistic structures (Steels, 2012b, p. 24).

Such complex adaptive systems are well-known for solving problems in a manner that is robust, flexible, adaptive and tailored to the environment. The aim of the language game paradigm is to achieve exactly these properties in the emergent communication systems.

**The Semiotic Cycle**

The *semiotic cycle* (Steels, 2012a) captures all processes that both the speaker and the listener go through during each language game or interaction. These are illustrated in Figure 2.1. The processes of the speaker are shown on the left side of the figure and those of the listener on the right side. Starting at the top, both agents first observe the environment in which they are situated, using their own (possibly different) sensors and construct their own (possibly different) world model. This process is called *grounding*. Afterwards, in a process called *conceptualisation*, the speaker determines the information that needs to be conveyed to the listener in order to successfully complete the communicative task. This information is captured in a semantic structure, i.e. the meaning, which is then expressed through a linguistic expression via a process called *production*. If the speaker does not possess adequate linguistic structures to complete production, it might need to *invent* them. The resulting utterance is passed from the speaker to the listener. The listener first analyses the utterance to obtain a semantic structure. This process is called *comprehension*. Afterwards, the listener *interprets* this semantic structure with respect to its own world model and performs an action that is relevant for completing the communicative task. This action may include pointing to a particular object, performing a gesture, moving an object, etc. Alternatively, the listener may signal that it does not understand the utterance, e.g. when the speaker has just invented a new linguistic expression or the listener has no adequate linguistic structures for completing comprehension.

As illustrated in Figure 2.1, the processes just described take place across three different levels. Specifically, grounding takes place on the sensorimotor level. Conceptualisation and interpretation take place on the conceptual level and production and comprehension take place on the language level.
2.2. LANGUAGE GAMES WITH BABEL

Figure 2.1: The semiotic cycle captures the processes on the sensorimotor level, the conceptual level and the language level that both the speaker and the listener go through during each interaction in a language game experiment.

Learning through Feedback and Alignment

The final part of every language game interaction, not shown in the semiotic cycle, consists of feedback and learning. Specifically, the speaker checks the action performed by the listener to determine the outcome of the game (success or failure), and provides feedback to the listener. This feedback may consist of pointing to the object that was intended, performing the gesture that was expected, etc. If the listener had signalled that it did not understand the utterance, the process of adoption can start. Here, the listener expands its communication system based on the observed utterance, the speaker’s feedback, its own current semantic and linguistic structures and its world model. Otherwise, both agents engage in a process called alignment where they update their communication systems based on the outcome of the game such that these are closer to each other and better suited for future interactions. Alignment in language games is based on ample evidence from psycholinguistics that human interlocutors rapidly and unconsciously self-organise their communication systems on all levels during conversation (Garrod and Doherty, 1994; Pickering and Garrod, 2006; Steels, 2011b, 2012b, p. 23). The specific operationalisation of alignment depends on the particular language game experiment, but it is typically based on lateral inhibition\(^1\) (Steels, 1995) or statistical pre-emption (Goldberg, 2011; Boyd and Goldberg, 2011; Goldberg, 2019, Ch. 5). It involves rewarding linguistic structures that can be used successfully, while punishing competing structures and structures that cannot be used successfully in communication. Competition may arise, for example, when multiple linguistic forms express the same semantic structure or vice versa. Crucially, alignment of communication systems is tightly connected to the self-organisation of linguistic conventions within a population of agents (Steels, 2000b, 2012b, p. 23-24). Specifically, it creates a positive feedback loop between the usage and the success of particular linguistic structures, which in turn leads to more agents in the population aligning their communication systems.

---

\(^1\)Lateral inhibition was first proposed by Ernst Mach in the 1860’s to explain edge detection in the receptive fields of the human eye (Mach, 1865)
systems to these structures and ultimately to the conventionalisation of these structures within the population.

The main challenge in the operationalisation of a language game consists in finding adequate grounding, conceptualisation, production, invention, comprehension, interpretation, alignment and adoption operators that allow the population of agents to develop and converge on a shared language.

2.2.3 Language Game Experiments

The language game paradigm has been used in a large number of experiments to investigate the acquisition and emergence of vocabularies, concepts and, more recently, grammars. Together, these experiments convincingly show that populations of agents can converge on effective and efficient shared communication systems through local interactions, the selection, self-organisation, level-formation and reinforcement learning applied to semantic and linguistic structures, and a number of communicative pressures. An overview of these experiments is provided below, following the three waves of language game experiments as identified by Van Eecke (2018, p. 20-22).

Emergence of Vocabularies

The most canonical language game experiment is the Naming Game (Steels, 1995), where a population of agents develops a common vocabulary to refer to objects (or predefined meanings) using proper names. Since its inception, a large literature has accumulated around this game. For instance, agent-based models were used to replicate the Naming Game using physical robots (Vogt, 2000; Steels and Kaplan, 2000; Steels and Loetzsch, 2012), to study a variety of alignment mechanisms (Wellens, 2012) and to study the effect of intrinsic motivation and active learning mechanisms (Schueller and Oudeyer, 2016). Mathematical models have been used to determine scaling laws (Baronchelli et al., 2006), proof convergence under various conditions (De Vylder and Tuyls, 2006) and study the effects of population structure (Dall’Asta et al., 2006; Liu et al., 2009). The emergence of set of conventions in the form of the Naming Game has been so extensively studied that it can be considered a solved problem, and its solutions are re-used in more advanced experiments (Steels, 2011b).

Emergence of Concepts

In these language game experiments, the agents learn to coordinate their conceptual systems. These experiments extend the Naming Game in that the meanings underlying words are no longer predefined, but they are semantic concepts grounded in the agents’ sensori-motor experiences. For example, concepts may capture the colour of objects expressed in a continuous, multi-channel colour space, or an action to perform in terms of the angles of motors in robotic joints. The agents thus not only need to learn mappings between words
and semantic concepts, but simultaneously learn these semantic concepts. Because the sensorimotor observations of autonomous agents are not necessarily identical, e.g. due to different perspectives, different lighting conditions or different sensors, the shared language that emerges during these language games forms an abstraction layer over these heterogeneous observations. Concretely, the agents typically divide their space of sensorimotor observations in convex regions, each of which corresponds to a particular concept, and simultaneously establish a shared lexicon for each of these regions. These convex regions are not necessarily identical between the agents and competition may arise as multiple concepts become associated to a single word or vice versa. The coordination of conceptual systems thus requires two types of mechanisms: one for updating concepts such that they capture distinctions that are functional in the agents’ environment, and one for updating word-concept mappings. The first type of mechanism typically relies on discrimination and prototypicality (see e.g. Rosch (1973)), while the second type relies on the solutions from the Naming Game experiment.

These kinds of experiments have been conducted in various domains (Van Eecke, 2018, p. 21), such as colour (Belpaeme, 2002; Belpaeme and Bleys, 2005a; Steels and Belpaeme, 2005; Bleys and Steels, 2009), spatial relations (Loetzsch et al., 2008a; Spranger, 2012, 2013), action language (Steels et al., 2012) and vowel systems (de Boer, 2000; Oudeyer, 2001).

**Emergence of Grammar**

More recently, the focus of language game experiments has shifted towards the acquisition, emergence and evolution of grammar. These experiments are considerably more complex since the agents now need to learn mappings between larger semantic and morphosyntactic structures that may feature compositionality or hierarchy. In general, previous experiments on the emergence of grammar have followed one of two approaches that will be discussed below. Additionally, mechanisms that enhance the agents’ autonomy, robustness and flexibility have been investigated in the context of grammar learning experiments. Examples include re-entrance (Steels, 2003; Van Eecke, 2015), where the speaker creates a model of the listener to detect possible ambiguities, and intrinsic motivation (Steels, 2004a; Steels and Wellens, 2007; Cornudella Gaya, 2017), where the agents gauge the complexity of their own communicative interactions and set up a learning trajectory.

The first approach focusses on semantic strategies to dampen referential ambiguities. In these experiments, the agents learn compositional semantic structures that capture relevant distinctions in the world. Additionally, they learn grammatical structures that convey information about these semantic structures, e.g. how semantic concepts interact by structuring corresponding lexical items in an utterance. The grammatical structure provides cues for the listener in order to reduce the possible interpretations of this utterance or, put differently, dampen the referential ambiguity. The main focus of these approaches lies on the semantic aspect, namely on learning semantic concepts and the different con-
ceptualisation strategies that can be used to obtain the compositional semantic structures. This approach has been applied to various domains (Van Eecke, 2018, p. 22), such as colour (Bleys and Steels, 2009; Bleys, 2016), spatial relations (Spranger et al., 2010; Spranger and Steels, 2015; Spranger, 2015, 2016, 2017), quantifiers (Pauw and Hilferty, 2012) and logical operators (Sierra-Santibáñez, 2014; Sierra Santibáñez, 2018).

In the second approach, morpho-syntactic strategies are used to dampen the referential ambiguity. Specifically, these experiments focus on the emergence and evolution of word order or markers, such as case, gender or number, to indicate which words in the utterance belong together, thereby again reducing the ambiguity and the cognitive effort for the listener. This approach has been applied to case systems (Steels et al., 2012; van Trijp, 2013, 2016; Lestrade, 2016), agreement (Beuls et al., 2010; Beuls and Steels, 2013), word order (Steels and Casademont, 2015; Garcia-Casademont and Steels, 2016), Russian aspect (Gerasymova and Spranger, 2010, 2012), and the co-emergence of word order and grammatical categories (Van Eecke, 2018, Ch. 7).

The experiments presented later on in this dissertation contribute, on the one hand, to the literature on the acquisition of concepts (Chapters 4), and on the other hand, to the literature on the acquisition of grammar (Chapter 6). These experiments will push forward the state of the art by doing away with certain scaffolds that were used in previous experiments, and by operating on a larger scale in terms of the complexity of the input and the linguistic phenomena considered.

2.2.4 Babel

As discussed in Section 2.2.2, operationalising language game experiments requires implementing the various processes of the semiotic cycle (see Figure 2.1) in a multi-agent setting. While there are a number of software packages available, not all of them are suitable for the language game paradigm. NetLogo (Wilensky, 1999), for example, is a well-known platform that allows to model a variety of multi-agent systems from a complex systems science perspective. It contains examples of multi-agent systems, such as predator-prey systems, bird flocking, or ant colonies. However, it focuses solely on setting up a multi-agent architecture and does not contain sufficient functionality for setting up processes on the conceptual or linguistic level. Alternatively, NaminggamesAL (Schueller, 2018) and MoLE (Modelling Language Evolution) (Lestrade, 2017) offer more functionality that is targeted towards language game experiments. However, the former can only be used for setting up naming games in simulated environments, while the latter focuses exclusively on the language level and in particular on the emergence of case systems (Lestrade, 2016). Both of these tools thus lack the ability to set up language game experiments that make use of all three levels described in the semiotic cycle. Finally, EGG (Kharitonov et al., 2019) is a framework for setting up multi-agent emergent communication scenarios. Because of its origins in the MARL research community, it suffers from many of the shortcomings
discussed in Section 2.2.1. To illustrate, the EGG framework provides several neural architectures, such as RNNs, GRUs, LSTMs or Transformers, to operationalise the ‘communication channels’ of the agents. These architectures, however, do not support bidirectional communication. Furthermore, the framework only allows to specify agents as either being ‘senders’ or ‘receivers’. Hence, the agents cannot switch their discourse roles. The Babel software package has been specifically designed to set up language game experiments including all processes of the semiotic cycle. Since its inception in 1998 (McIntyre, 1998), Babel has gone through several revisions (Loetzsch et al., 2008b; Steels and Loetzsch, 2010; Nevens et al., 2019b). In the following paragraphs, I provide an overview of the software components that are currently integrated in Babel.

Multi-Agent Architecture

A language game experiment is a multi-agent system. Babel’s EXPERIMENT-FRAMEWORK provides the necessary abstractions for creating a population of agents, keeping track of the population structure, selecting agents to participate in an interaction and assigning them the discourse roles of speaker or listener. By default, the EXPERIMENT-FRAMEWORK structures the population as a fully-connected network, randomly selects two agents for every interaction and randomly chooses one to be the speaker and the other to be the listener. However, all of these aspects are fully customizable. Further, the EXPERIMENT-FRAMEWORK provides a generic ‘interact’ function that allows to specify the interaction script of the language game in terms of the speaker and the listener, which have already been determined by the framework. Finally, the framework provides functionalities for running multiple experimental runs, either in series or in parallel.

Sensorimotor Level

The agents’ action and perception capabilities can be handled either through simulation or through embodiment using robots. In past experiments, Babel has been used with Sony AIBO dog-like robots (Steels and Kaplan, 2000), Sony QRIO humanoid robots (Spranger et al., 2012a), MYON humanoid robots (Hild et al., 2012; Steels et al., 2012), and the PERACT vision system (van Trijp, 2016, Ch. 3). Later, Nevens et al. (2019b) introduced a standardized interface for connecting Babel to robotic platforms through the ROBOT-INTERFACE package. This package includes a standard set of functions that are useful for implementing grounded language game experiments. Examples include scanning the robot’s environment, speaking, listening and pointing. These functions are provided in a hardware-independent manner, allowing Babel users to abstract away over the specific implementations of the robotic platforms and allowing Babel developers to easily add new robotic platforms by mapping specific hardware instructions to these high-level functions. Nevens et al. (2019b) demonstrated the ROBOT-INTERFACE package by integrating the Nao humanoid robot (Gouaillier et al., 2008) in Babel and using it in a didactic colour naming game experiment.
CHAPTER 2. BACKGROUND AND TECHNICAL FOUNDATIONS

Conceptual Level

INCREMENTAL RECRUITMENT LANGUAGE (IRL) is used to bridge the gap between the agents’ world model and the semantic structures that need to be expressed by the speaker or interpreted by the listener. When using IRL, the agents’ semantic structures are expressed in terms of procedural semantics. Section 2.4 is dedicated to this system.

Language Level

FLUID CONSTRUCTION GRAMMAR (FCG) is responsible for mapping between semantic structures and linguistic utterances through constructional language processing. The agents’ linguistic knowledge is captured in the form of constructions, which are conventionalised form-meaning mappings. Section 2.3 is dedicated to this system.

Experiment Monitoring

During a language game experiment, the MONITORS and PLOT-RAW-DATA frameworks can be used to keep track of experimental parameters, logging data and evaluation metrics, export them to data files and create graphs. The MONITORS framework uses an event-based architecture that cleanly separates the game’s interaction script and processes from the monitoring, recording and exporting of data. A number of useful events are provided by default, e.g. at the start and end of every interaction or at the start and end of a series of interactions. The framework can be easily used to track experimental parameters or collect metrics at various points throughout the experiment by specifying new events and event handlers. The PLOT-RAW-DATA framework is used to display the collected data in real time using dynamically updating graphs or to create static graphs after the experimental runs. Line plots or bar plots can be generated in order to compare different metrics from one experiment or to compare the same metric across differently configured experimental runs. Multiple runs are automatically averaged and statistics, such as standard deviation or percentiles, can be added.

In sum, the Babel software package offers all the necessary tools for setting up language game experiments, implementing all processes of the semiotic cycle and aggregating the results. These tools are highly configurable, modular and available through an open-source license via https://emergent-languages.org. All experiments presented later on in this dissertation are implemented using Babel.

2.2.5 Meta-Layer Architecture

An integral part of the Babel software package is its meta-layer architecture. Through this architecture, two layers of processing can be discerned: routine processing and meta-level processing. As noted by Doumen et al. (forthcoming), this separation follows insights from cognitive architectures (Laird et al., 1987; Maes and Nardi, 1988), cognitive science
2.2. LANGUAGE GAMES WITH BABEL

(Evans, 2003; Kahneman, 2011) and neurolinguistics (Osterhout and Holcomb, 1992; Hagoort et al., 1993). Routine processing deals with what is already known and can therefore be implemented efficiently, while the less efficient meta-level deals with problem solving and learning. This is achieved by the integration of the following components (Beuls et al., 2012; Van Eecke and Beuls, 2017; Van Eecke, 2018, p. 51-52):

- **Diagnostics** constantly monitor the processes going on during routine processing. If one of the diagnostic tests fails, a problem is created. The problem holds information about the process that went wrong.

- **Repairs** are methods that will try to solve problems. Depending on the type of problem that was diagnosed, specific repairs will be triggered. When a repair can solve the problem, it returns a fix. Using that fix, routine processing continues. When a repair cannot generate a fix, other repairs that specialise on the diagnosed problem can still be tried.

- **Consolidation strategies** are used at the end of an interaction. Specifically, when the fix turns out to be successful, it can be consolidated into the agent’s memory. This fix is then available for routine processing in later interactions. When the fix did not lead to a successful interaction, it can be discarded again.

In language game experiments, the meta-layer architecture can be used on three different levels (Beuls et al., 2012; Van Eecke, 2018, p. 52-53):

- **Language Level.** On this level, diagnostics and repairs operate over linguistic processing in terms of constructions. Specifically, diagnostics are ran after every construction application (see Section 2.3.4) and repairs typically produce fixes by means of new constructions. For example, fixes can produce constructions that cover part of the input that was not covered before. If this construction leads to a successful comprehension or production process, it is consolidated for later re-use. The meta-layer architecture on the language level is illustrated in Figure 2.2. The meta-layer architecture is most tightly integrated with the processes on the language level. Concretely, initial versions of this integration were presented by Steels and van Trijp (2011); Beuls et al. (2012) and van Trijp (2012). Later, a standard library of diagnostics and repairs was integrated in Fluid Construction Grammar (Van Eecke and Beuls, 2017; Van Eecke, 2018).

- **Process Level.** On this level, diagnostics and repairs operate on the processes of the semiotic cycle (see Figure 2.1). Concretely, problems can be diagnosed and repaired during grounding, conceptualisation, production, comprehension or interpretation. Note that these diagnostics and repairs operate over comprehension and production as a whole, while the previous level of diagnostic and repairs are active within these processes on the level of construction application. An example of a problem on the process level is that the speaker does not know an utterance that can express the
semantic structure it just conceptualised. The process of invention then constitutes a repair, e.g. by creating one or more new constructions that express the semantic structure.

- **Agent Level.** On this level, diagnostics and repairs operate over multiple processes of the semiotic cycle at once. The prime example of this level of diagnostics and repairs is re-entrance (Steels, 2003; Van Eecke, 2015), where the speaker creates a model of the listener to detect possible ambiguities. This requires the speaker to run both comprehension and interpretation as a diagnostic, as if it were the listener, and to restart its conceptualisation and production processes as a repair.

![Figure 2.2: Meta-layer architecture illustrated on the language level. Diagnostic are ran after every construction (cxn) application. Fixes consist of new constructions, which can be consolidated into the agent’s memory (cxn inventory). Image from Van Eecke and Beuls (2017)](image)

The separation between routine processing and meta-layer processing allows the agents to deal with problem solving and learning on the fly and ensures both robustness and open-endedness. In other words, it allows the agents to keep on learning and adapting indefinitely, which is a highly desirable property of intelligent systems. It also removes the distinction between a training phase and an operational phase, as is typically found in many other learning systems.

### 2.3 Computational Construction Grammar with FCG

This section introduces Fluid Construction Grammar (FCG) (Steels, 2011a, 2017; van Trijp et al., 2022, https://fcg-net.org), the most advanced computational construction grammar formalism. I start this section by providing an overview of the basic tenets of construction grammar, the challenges and opportunities of computational construction grammar and a number of computational construction grammar formalisms (Section 2.3.1). Next, I discuss the main ideas underlying FCG (Section 2.3.3) and how these can be operationalised through the building blocks of FCG (Section 2.3.4). Afterwards, I zoom in on a recent addition to FCG that will be relevant later on in this dissertation, particularly in Chapter 6. This concerns the integration of a categorial network for modelling the emergence and evolution of grammatical categories (Section 2.3.5). Finally, I present a novel contribution
to FCG that uses recurrent neural networks for providing search heuristics that substantially optimise the search process involved in large-scale construction grammars (Section 2.3.6).

### 2.3.1 Construction Grammar

The research field of construction grammar (CxG) aims to develop an all-encompassing scientific theory of language, from the representations of linguistic structures in the brain to their usage, acquisition and evolution. In construction grammar, all linguistic knowledge is captured in terms of constructions. These are bidirectional mappings between any kind of form, i.e. phonology, morphology, syntax, prosody, etc., and any kind of meaning, i.e. semantics and pragmatics. Construction grammar abandons the generative constituent structure grammar approach (Chomsky, 1965) that focusses mostly on syntax. This approach can also be called the dictionary-and-grammar approach (Taylor, 2012, p. 8), as linguistic knowledge is captured as a dictionary of words together with grammar rules for combining those words. While the list of grammar rules should suffice to capture every possible utterance, the dictionary-and-grammar approach both overgenerates, i.e. produces non-grammatical utterances, and undergenerates, i.e. fails to produce certain grammatical utterances. There are, for example, utterances that are not compositional on either the form side, such as ‘many a year’, or the meaning side, such as ‘kick the bucket’, and thus fall outside of the grammar rules. A possible solution is to keep a list of exceptions that are still grammatical but not covered by the grammar rules. However, Hilpert (2014, p. 7) notes that in fact most utterances have non-compositional aspects to them, making the list of exceptions far exceed the list of grammar rules. As a result, the list of grammar rules covers only a narrow subset of actual language use. Therefore, construction grammar advocates for using the same machinery to model both the compositional and non-compositional aspects of language (Fillmore, 1988, p. 534). This machinery is, as mentioned, a set of constructions.

#### Basic Tenets

The basic tenets of construction grammar were laid out by among others Fillmore (1988); Goldberg (1995); Kay and Fillmore (1999); Croft (2001) and Goldberg (2006), and can be summarized in the following four points (van Trijp et al., 2022; Van Eecke, 2018, p. 28):

- All linguistic knowledge is captured by **conventionalised form-meaning mappings**, or **constructions**. What exactly constitutes form and meaning can be broadly interpreted. Everything that is provided as input to the comprehension process can be considered form, e.g. phonology, morphology, syntax, prosody, intonation, etc. Likewise, everything that is used as input for production can be considered meaning, e.g. semantics and pragmatics.

- All constructions are situated somewhere on the **lexicon-grammar continuum** as
no distinction is made between words and grammar rules. This continuum stretches from fully concrete constructions, mapping a specific form to a specific meaning, to fully abstract constructions, such as argument structure constructions.

- Constructions cut through all layers of linguistic analysis. Instead of analysing different layers, such as morphology, phonology, syntax, semantics and pragmatics, separately, constructions may contain information from any of these layers at the same time.

- Construction grammars are dynamic systems. They capture the linguistic knowledge of an individual. This constantly changes as constructions become more or less entrenched through language use or new constructions are learned.

**Computational Construction Grammar**

The aim of computational construction grammar (CCxG) is to operationalise these basic tenets, and to formalise insights and analyses from construction grammar into concrete processing models (Van Eecke and Beuls, 2018). There are five main reasons for operationalising construction grammars in computational models (Steels, 2017; van Trijp et al., 2022).

- For CxG to become an all-encompassing theory of language, this requires the modelling of thousands of constructions and their intricate interactions. This quickly becomes intractable to do by hand. Computational construction grammar allows to automatically verify (large-scale) construction grammars in terms of consistency and preciseness.

- Computational construction grammar facilitates corpus studies. This allows scholars to automatically track the presence of constructions in texts or to verify the coverage of a particular grammar.

- Computational construction grammar facilitates the standardisation of construction grammar models by providing a single representation and processing mechanism for constructions that is shared across the community. This allows researchers to compare, exchange and integrate their findings more easily.

- Computational construction grammar allows to integrate findings from construction grammar in other domains of linguistics where computational techniques are used, e.g. in historical linguistics, language acquisition and language evolution.

- Computational construction grammar allows to integrate findings from construction grammar in language technology applications, such as visual question answering (Nevens et al., 2019a), visual dialogue (Verheyen et al., 2021) and semantic frame extraction (Willaert et al., 2020; Beuls et al., 2021; Willaert et al., 2021, 2022).
Next to the opportunities that computational construction grammar brings, operationalising concrete processing models of construction grammar also involves a number of challenges. Specifically, it requires (i) a formalism to define constructions in a machine-readable format, (ii) a processing engine for applying these constructions in both the comprehension and production direction, and (iii) a computational representation of form and meaning. The first two challenges have been tackled by several construction grammar formalisms. In the following section, I provide an overview of these formalisms. For the third challenge, namely a computational representation of form and meaning, several options are available. Linguistic forms can be represented using speech sounds or text. Typically, they are represented using strings, i.e. sequences of characters, that capture single characters, syllables or words. For representing meaning, a variety of semantic representations are available. Among others, one can use predicate logic, frame semantics (Fillmore and Baker, 2001), Abstract Meaning Representation (AMR) (Banarescu et al., 2013), or procedural semantics (Woods, 1968; Winograd, 1972; Johnson-Laird, 1977).

**Grammar Formalisms**

Several computational construction grammar formalisms have been developed, but not all of them satisfyingly address the challenges posed in the previous paragraph (van Trijp et al., 2022; Van Eecke, 2018, p. 28). Embodied Construction Grammar (ECG) (Bergen and Chang, 2005; Feldman et al., 2009) maps forms onto conceptual representations that parametrize mental simulations. In ECG, “understanding an utterance thus involves at least two distinct processes: analysis to determine which constructions the utterance instantiates, and simulation according to the parameters specified by those constructions.” (Bergen and Chang, 2005). However, ECG only supports comprehension and not production. Dynamic Construction Grammar (DCG) (Dominey et al., 2017) uses neural networks to find patterns in mappings between sentences and their argument structure. Because of this black-box architecture, there is no explicit formalism to define constructions in DCG. Similarly, Template Construction Grammar (TCG) (Barrès and Lee, 2014; Barrès, 2017) uses neural networks to study the interaction between vision and language. For the same reason as DCG, it thereby lacks a formalism to define constructions. Sign-Based Construction Grammar (SBCG) (Boas and Sag, 2012) integrates ideas from construction grammar into Head-Driven Phrase Structure Grammar (HPSG), which has its origins in generative grammar. SBCG thereby does not completely embrace the basic tenets of construction grammar. Finally, Fluid Construction Grammar (Steels, 2011a, 2017; van Trijp et al., 2022) offers an open-ended construction grammar formalism and processing engine that supports both comprehension and production. FCG stays close to the basic tenets of construction grammar and aims to be as theory-neutral as possible with respect to the constructional analysis that is being implemented. Instead of being a particular theory or a specific grammar, FCG can be thought of as a special-purpose programming language that provides all the necessary building blocks for operationalising any constructionist approach to language (van Trijp
For a more in depth discussion on differences and similarities between computational construction grammar formalisms, I refer to Chang et al. (2012) for ECG-FCG and van Trijp (2013) for SBCG-FCG.

In the context of this dissertation, FCG is well suited to operationalise the language processing capabilities of autonomous agents because of the computational operationalisation of the basic tenets of construction grammar. Indeed, constructions offer a tight integration between form and meaning which is crucial for implementing the basic function of language, namely to produce utterances as a speaker (i.e. map meanings to forms) and to comprehend utterances as a listener (i.e. map forms to meanings). Furthermore, by being a dynamic, open-ended system, constructions can be added to FCG at any point in time, which is crucial for implementing language game experiments on the emergence and acquisition of grammar, and for learning generalisations of constructions somewhere on the lexicon-grammar continuum. Finally, by including all layers of linguistic analysis in constructions, autonomous agents can use them for solving commutative problems.

### 2.3.2 Fluid Construction Grammar

Over its 20 years of development, Fluid Construction Grammar (FCG) (Steels, 2011a, 2017; van Trijp et al., 2022, https://fcg-net.org) has become the most advanced and feature-rich computational construction grammar formalism. Being part of the Babel toolkit, FCG was initially developed for experiments on the emergence of grammar (see e.g. Steels (2004b); De Beule and Bergen (2006); Bleys (2008); Beuls and Höfer (2011); Gerasymova et al. (2012); Pauw and Hilferty (2012) and Steels et al. (2012)). Given these origins, FCG relies on many of the same principles and ideas as found in the language game paradigm. For instance, language processing should be bidirectional as this is required to model both the speaker’s and the listener’s processes in the semiotic cycle. Further, constructions in FCG typically keep a score, such that agents can self-organise their linguistic systems through variation, selection and reinforcement learning over constructions. Apart from evolution experiments, FCG was also used for case studies in linguistics from a construction grammar perspective (see e.g. van Trijp (2011); Beuls (2012) and van Trijp (2014)). A new high-level notation for FCG was introduced around 2015 and is described in Steels (2017) and Van Eecke (2018, Ch. 3). This notation is easier to learn and use for evolutionary linguists and construction grammarians alike and has quickly become the standard. Since then, FCG has been used in more advanced experiments on the emergence of grammar (see e.g. Cornudella Gaya et al. (2016); Garcia-Casademont and Steels (2016), Van Eecke (2018, Ch. 7) and Doumen et al. (forthcoming)), linguistic case studies for various natural languages (see e.g. Marques and Beuls (2016); Beuls et al. (2017); Beuls (2017); Van Eecke (2017)), a range of language technology applications (see e.g. Nevens et al. (2019a); Verheyen et al. (2021); Beuls et al. (2021); Willaert et al. (2020, 2021, 2022)) and an exploratory study on
computational creativity (Van Eecke and Beuls, 2018). The high-level notation of FCG also facilitated the introduction of more advanced features, such as a library of general purpose learning operators through pro- and anti-unification of constructions that is integrated in the meta-layer architecture (Van Eecke and Beuls, 2017; Van Eecke, 2018; Steels et al., 2022), visualization of constructional dependencies (Hoorens et al., 2017), a categorial network for modelling emergent grammatical categories (Van Eecke, 2018; Steels et al., 2022) (see Section 2.3.5), and advanced search heuristics for large-scale grammars (see Section 2.3.6). Finally, FCG is not only integrated in the Babel software package, it is also made available through a stand-alone integrated development environment: the FCG Editor (van Trijp et al., 2022).

### 2.3.3 Language Processing as Problem Solving

The main idea underlying FCG is that it treats constructional language processing as a problem solving process (Bleys et al., 2011; Van Eecke and Beuls, 2017; Steels and Eecke, 2018). Following Van Eecke (2018, p. 33) and Steels and Eecke (2018), I will first lay out the basic components of a state-space problem as defined by Russell and Norvig (2009, p. 66), illustrated via the sliding block puzzle of size eight, or the eight-puzzle (Russell and Norvig, 2009, p. 70), and then map these basic components to constructional language processing and FCG in particular.

The eight-puzzle consists of a 3x3 grid with eight sequentially numbered tiles and one free space. A tile that is adjacent to the free space can slide into it, creating a new free space. Starting from a random configuration of tiles, such as the one shown in Figure 2.3a, the goal is to reach the state depicted in Figure 2.3b.

![Figure 2.3: An instance of the sliding block puzzle of size eight.](image)

The problem posed by the eight-puzzle can be defined by the following four components:

- **The state representation** captures all relevant information about the problem at a certain point in time in a representation that can be computationally manipulated. For the eight-puzzle problem, the state representation captures the current configuration of tiles, e.g. in an array data structure.
• **The initial state** is the state representation before any problem solving has started. For the eight-puzzle problem, this can be any randomly selected configuration of tiles, such as the one in Figure 2.3a.

• **The operators** are actions that can be applied to a state representation and by doing so create a new state representation. For the eight-puzzle problem, this involves sliding a tile adjacent to the blank space into the blank space. The new state consists of the configuration of tiles obtained by this move.

• **The goal test** is a function that decides whether the current state representation is a solution to the problem. For the eight-puzzle problem, the configuration depicted in Figure 2.3b is the only one that qualifies.

A problem solving process is a process that, starting from the initial state representation, applies operators to state representations until the goal test is satisfied. This constitutes a search process of finding the sequence of operators that can transform the initial state into the goal state. Certain sequences of operators might lead to a dead end, requiring the search process to backtrack. When multiple operators are possible at every state, as is the case in the eight-puzzle, a search space is formed. Such search spaces can grow very large very rapidly. Therefore, heuristics are commonly used to traverse the search space in an informed way, i.e. deciding which operator to apply in a given state.

The four components of a state-space problem, illustrated through the eight-puzzle above, can be mapped to constructional language processing, and in particular to FCG, as follows (Van Eecke and Beuls, 2017; Steels and Eecke, 2018; Van Eecke, 2018, p. 34-35):

• **The state representation** is called the *transient structure*. It holds all information that is known up to the current point in processing the utterance in comprehension or the meaning representation in production. Represented as a feature structure, it can hold any kind of linguistic information, including syntactic, semantic, pragmatic, morphological, prosodic, phonological, phonetic and multi-modal information.

• **The initial state** is the *initial transient structure*. It represents the input to the comprehension or production process as a feature structure. A process called *de-rendering* transforms the input into a feature structure. In comprehension, strings and ordering constraints between them are typically extracted from the input utterance. In formulation, the input meaning is typically represented as a set of predicates, which can be directly captured as a feature structure.

• **The operators** are the *constructions*. Constructions *apply* to transient structures and in doing so create new transient structures. Specifically, a construction specifies a set of *pre-conditions* and *post-conditions*. When a construction’s pre-conditions are satisfied by the current transient structure, the construction applies and manipulates
the transient structure through its post-conditions. The constructions are stored in a *construction inventory*.

- **The goal test** checks after each construction application if the current transient structure is a solution. What exactly constitutes a solution differs depending on the direction of processing. Typical goal tests for comprehension and production are discussed later on.

The search process faced by FCG consists in finding the sequence of constructions to apply in order to transform the initial transient structure into a transient structure that qualifies as a solution, and ultimately to map an utterance onto its meaning representation or vice versa. This search process is illustrated in Figure 2.4. From a theoretical perspective, construction grammar allows for the free combination of constructions (Goldberg, 2006, p. 22), i.e. allowing any construction to apply at any point in time as long as its pre-conditions are satisfied. Consequently, multiple constructions could apply at any time and thus finding this sequence of construction applications in FCG creates a search space. This search space is combinatorial in nature and becomes intractable as the construction inventory grows in size. Therefore, heuristics that determine which branch to explore and which construction to apply are necessary.

![Figure 2.4: Schematic representation of FCG’s search process. Construction application leads to the creation of new transient structures. When the transient structure turns out to be a dead end, backtracking is required, exploring other construction applications. The goal test is applied to every transient structure to check for a solution. Image from Van Eecke (2018, p. 36)](image-url)
2.3.4 Building Blocks of FCG

The building blocks of FCG that were mentioned in the previous section, such as *(initial)* transient structures, constructions, construction application, construction inventory, goal tests, etc., will be described in greater detail in the following paragraphs. Throughout this section, I will use a small grammar as a running example. This grammar allows to map the utterance "The linguist likes the mouse" onto a meaning representation in first-order logic and vice versa\(^2\). The meaning representation for this utterance is \{\text{MOUSE}(M), \text{UNIQUE}(M), \text{LINGUIST}(L), \text{UNIQUE}(L), \text{DEEP-AFFECION}(L,M)\}. This meaning representation is visualized as a semantic network in Figure 2.9.

**Transient Structures**

The transient structure is a data structure that holds all information up to the current point in processing. Concretely, the transient structure is a feature structure that consists of a number of *units*, each with a *unit name* and a *unit body*. The unit body consists of a set of feature-value pairs. Unit names and feature names are constants. Feature values can be of different types: symbols, logic variables (symbols preceded by a question mark), sequences, sets, sets of predicates, sequences of predicates or again feature-value pairs. The feature structures used in FCG are not typed and the set of possible features that can be used is completely open-ended. FCG does not know about nouns, articles, noun phrases, mice or linguists. The only "meaning" that is assigned to the symbols in FCG is how these symbols are used within the grammar.

![Figure 2.5: An example transient structure containing four units: 'root', 'noun-phrase-19', 'the-9', and 'mouse-6'. The 'constituents' feature is used to draw the units in a tree structure.](image)

Figure 2.5 shows an example transient structure. It contains four units: ‘root’, ‘noun-phrase-19’, ‘the-9’, and ‘mouse-6’. The ‘constituents’ feature is used to draw the units in a tree structure.

\(^2\)The same grammar can be used to map between “The *mouse* likes the linguist” and its meaning representation. This grammar is included by default in FCG and the FCG Editor (van Trijp et al., 2022).
phrase-19’ , ‘the-9’, and ‘mouse-6’. The ‘root’ unit is treated differently from the other units and will be discussed later on. The other units contain feature-value pairs of varying types. These can be distinguished from each other by their notation. For instance, the ‘args’ feature is a set, denoted by the square brackets. The ‘sem-cat’ feature has another feature-value pair as its value, denoted by the indentation. Finally, both the ‘form’ and the ‘meaning’ feature have a set of predicates as their value, indicated by the curly braces.

The units in Figure 2.5 are visualised as a tree data structure. This is purely for visualization purposes. In this example, FCG uses the ‘constituents’ feature, specified as a set of values in the ‘noun-phrase-19’ unit, to draw the units as such. This is, however, completely configurable. Another feature can be specified to control the structuring of units within the transient structure, e.g. to highlight constituency or dependency structure.

**Initial Transient Structure**

The initial transient structure constitutes the start of the search process. The only information it can hold at this point is a formal representation of the input that was provided to the comprehension or production process. A process called *de-rendering* transforms this input into a feature structure. Specifically, it will store the input as a feature structure in a special unit called ‘root’.

![Initial Transient Structure](image)

(a) Comprehension

(b) Production

Figure 2.6: The initial transient structure for the demo grammar in both the comprehension and production direction.

In comprehension, the input of our running example is the utterance “The linguist likes
the mouse”. The result of the default de-rendering process is shown in Figure 2.6a. The input string is first tokenized and each token is given a unique identifier. The list of tokens and ordering relations between them are then transformed into predicates that are stored under the ‘form’ feature. Specifically, the ‘string’ predicate links tokens to identifiers, the ‘meets’ predicate denotes adjacency, the ‘precedes’ predicate captures precedences and the ‘sequence’ predicate captures the entire list of tokens. Other de-renderers are available, e.g. for only capturing ‘string’ and ‘meets’ predicates.

In production, the input in the example grammar is a set of predicates in first-order logic. This can be transformed into a feature-value pair in a straightforward manner, since a set of predicates is a supported feature type in FCG. The de-rendering process simply takes the set of predicates and places them under the feature ‘meaning’ in the ‘root’ unit. The result is shown in Figure 2.6b.

Constructions

Constructions are at the heart of FCG. They are the search operators that allow FCG to transform an initial transient structure into a solution transient structure, and ultimately to map an utterance onto a meaning representation or vice versa. Crucially, the same constructions are used both for comprehension and production. Before diving into the process of construction application, I first introduce the design of the constructions that makes this bidirectional processing feasible.

![Figure 2.7](image-url)

Figure 2.7: Schematic representation of a construction. Constructions consist of a contributing part, left of the arrow, and a conditional part, right of the arrow. Units on the conditional part are split in comprehension locks and production locks (also called formulation locks). Image from Van Eecke (2018, p. 41).

Figure 2.7 shows a schematic representation of a construction. Just like transient structures, constructions are feature structures that consist of units. In contrast to transient structures, the unit names in constructions are variables. A construction groups a number of units under a particular name, such as ‘example-cxn’ in Figure 2.7, and structures them in a particular way. Concretely, a construction consists of two parts: the conditional part and the contributing part. The conditional part is on the right side of the arrow and the contributing part on the left side.
The conditional part specifies the construction’s pre-conditions and consists of one or more units. The pre-conditions differ depending on the direction of processing, mainly because different information is present in the transient structure. This could already be observed at the initial transient structure in Figure 2.6. Therefore, units on the conditional part are again split in two: comprehension locks and production locks. The former specify the construction’s pre-conditions in comprehension and the latter specify the pre-conditions in production. The feature-value pairs of the comprehension lock are written below the line, and those of the production lock above the line. In Figure 2.7, the conditional part consists of two units: ‘?unit-1’ and ‘?unit-2’.

The contributing part specifies the construction’s post-conditions and consists of zero or more units. These do not differ depending on the direction of processing. The schematic construction in Figure 2.7 has two units on the contributing part: ‘?unit-1’ and ‘?unit-3’.

Construction Application

Construction application consists of matching and merging. First, a construction checks if its pre-conditions match with the transient structure. Matching is a unification-based process that succeeds when the active lock\(^3\) of every unit in the construction’s conditional part can unify with a unit in the transient structure. When matching succeeds, merging begins. Merging is another unification-based process that succeeds when both the construction’s contributing part and the non-active locks of the conditional part can unify with units in the transient structure. Any information from the construction that was not yet in the transient structure is added by the merge. If no conflicts arise during merge, the construction has applied successfully.

Figure 2.8 illustrates the application of a single construction. This figure consists of the construction (cxn) itself (Figure 2.8b), the transient structure used for matching (Figure 2.8a) and the transient structure after merge (Figure 2.8c). In this example, the noun-phrase-cxn applies during comprehension. Hence, it’s pre-conditions for comprehension are highlighted in blue. These pre-conditions are met by unifying the cxn’s ‘?article’ unit with the ‘the-2’ unit in the transient structure and the cxn’s ‘?noun’ unit with the ‘mouse-2’ unit in the transient structure. During merge, the non-active locks of these units can also unify and the only new information is the ‘dependants’ feature specified in the cxn’s ‘?noun’ unit that is added to the ‘mouse-2’ unit in the transient structure. However, because the cxn’s ‘?article’ unit and ‘?noun’ unit specify the same variable ‘?x’ in the ‘args’ feature, these are also unified during merge, resulting in the same value for the ‘args’ feature in both the ‘the-2’ unit and the ‘mouse-2’ unit. These variables correspond to variables that are used in the ‘meaning’ features of these units, thereby connecting the corresponding first-order logic predicates.

\(^3\)Comprehension locks during comprehension and production locks during production.
The ‘?noun-phrase’ unit does not match with any unit in the transient structure. However, its pre-conditions contain the ‘hash’ special operator, indicated by the ‘#’ symbol. This operator is used to match on features specified in the ‘root’ unit. In Figure 2.8, the comprehension lock of the ‘?noun-phrase’ unit can indeed unify with ‘root’. During merge, the ‘hash’ operator will take these features out of the ‘root’ unit and place them in a new unit, in this case ‘noun-phrase-4’. It is through this mechanism that constructions process information from the input, both in comprehension and production.

The construction application in Figure 2.8 can be summarised as follows. The NOUN-PHRASE-CXN looks for something of type article, something of type noun and an adjacency relation between the forms that are associated to these units. This is applicable to ‘the-2’ and ‘mouse-2’. By its application, the construction creates something of type noun-phrase that captures the article and the noun and links together their meaning representations, such that they have the same referent ‘?x-85’. This creates a new transient structure that can again serve as the pre-conditions for other constructions.

Matching and merging are complex unification-based processes. In fact, FCG uses a number of matching and merging algorithms, depending on the feature type that is being unified. Furthermore, next to the ‘hash’ operator, several other operators are implemented in FCG. For example, operators that prevent constructions from applying an infinite number of times (called footprints) or allow for calling an arbitrary function during matching and merging of a particular feature (procedural attachment). A complete description of these features falls outside the scope of this chapter. I refer the reader to Van Eecke (2018, Ch. 3)
for a complete overview of FCG’s features and to Steels and De Beule (2006) and De Beule (2012) for a formal specification of matching and merging.

**Construction Inventory & Managing Search**

The *construction inventory* stores all constructions of a grammar and determines how they are organised. This can have a large impact on the processing efficiency of the grammar. By default, the construction inventory keeps all constructions in an unordered set and the FCG engine explores the search space in a depth-first manner. Specifically, the engine checks (in random order) which constructions can apply to the current transient structure and creates new transient structures for each successful construction application result. These transient structures are scored according to a heuristic, by default the depth in the search tree, and the transient structure with the highest score is further explored. This process continues until a solution is found, as explained in Section 2.3.3. As the grammar grows in size, depth-first exploration of the search space soon becomes intractable. Several mechanisms are available for steering the search process (Van Eecke, 2018, p.46-49). I discuss four of them:

- **Construction Sets.** Constructions can be structured in a number of ordered sets, e.g. as in Beuls (2011). First, constructions from the first set are tried. If no more constructions can apply, constructions from the second set are tried, and so on. The ordering of the sets depends on the direction of processing. For example, in comprehension, morphological constructions are typically tried before grammatical constructions. This allows the grammatical constructions to build on the morphological information from the input utterance. In production, these sets are typically tried in the reverse order such that the grammatical structure is built first and then instantiated with morphological elements. While construction sets can drastically reduce the search space, the idea of ordered sets is not in line with the free combination of constructions (Goldberg, 2006, p. 22), it becomes tedious to manually determine this kind of ordering relations between constructions in larger grammars, and it hinders the automatic learning of construction grammars (Van Eecke et al., subm).

- **Construction Networks.** Constructions can be organised in priming networks (Wellens and De Beule, 2010; Wellens, 2011). Organising constructions in a network can be motivated from three different perspectives. First, taxonomic links or other types of relations between constructions can be an integral part of construction grammar theories, e.g. family relations (Goldberg, 1995, Ch. 3) or priming networks (Diessel, 2019). Second, relations between constructions can also emerge from a usage-based perspective, e.g. through schematicity (Langacker, 2000) or co-occurrences of syntactic patterns (Safran, 2001). Third, priming networks can be used to operationalise the psychological phenomenon of priming where past observations non-consciously influence current behaviour (Schacter and Buckner, 1998).
In FCG, priming networks can be used to operationalise either of these three perspectives. In terms of the latter, priming networks capture frequent co-occurrences of constructions by keeping weighted, directed links between them. The construction inventory keeps separate priming networks for each direction of processing. During constructional processing, when a construction applies, the constructions that are primed by this application can be found via the outgoing links of the applied construction in the network. The primed constructions are tried in the order of their priming strength. These strengths can be learned from a corpus or emerge in a usage-based manner during a language game, specifically by strengthening the links between co-occurring constructions in successful comprehension or production processes. Priming networks are one of the most efficacious ways of reducing the search space during constructional processing and can be learned in a straightforward manner. The downsides of priming, however, are that it relies only on local information, i.e. co-occurrences of constructions, and that the priming network becomes sparse and inefficient when longer-distance links are added (Van Eecke et al., subm).

- **Hashing Constructions.** The hashing of constructions, introduced by Van Eecke (2018, p. 48), can optimise the search process involved for morphological and lexical constructions, which typically make up the largest part of many grammars. These types of constructions typically match on information from the input, making their pre-conditions rather straightforward. Specifically, in comprehension, these constructions match on strings, while in production, they match on meaning predicates. Processing of these constructions can be optimised by keeping two hash tables: one for comprehension and one for production. The former uses strings as keys and the latter uses meaning predicates as keys. The values of these keys are the constructions that match on these strings or predicates, respectively. Given the strings or predicates present in the ‘root’ of the current transient structure, the FCG engine can now retrieve the set of applicable morphological and lexical constructions in constant time, without running the computationally expensive (unification-based) matching algorithm. When the ‘root’ is empty, the large set of morphological and lexical constructions is no longer considered for matching.

- **Scoring Constructions.** A best-first exploration of the search space can be obtained by relying on the scores of constructions. This strategy is typically used in evolutionary experiments where the scores of constructions reflect their entrenchment (Langacker, 1987; Schmid, 2007; De Smet, 2017; Theakston, 2017). Put differently, the score of a construction reflects how confident the agent is that this construction will lead to successful communication. In these experiments, the scores are typically updated through lateral inhibition dynamics (Steels, 1995) (see also Section 2.2.2).
2.3. COMPUTATIONAL CONSTRUCTION GRAMMAR WITH FCG

Meaning Representations

FCG does not impose the use of one particular kind of meaning representation. By default, it can handle any kind of meaning representation that consists of predicates that share their arguments by linking variables, or any kind of meaning representation that can be transformed into such predicates. Commonly used meaning representations in FCG include first-order logic (as in the example grammar used in this section), frame semantics (Fillmore, 1976) (as in Beuls et al. (2021)), Abstract Meaning Representation (AMR) (Banarescu et al., 2013), and procedural cognitive semantics formalised through the Incremental Recruitment Language (IRL) system (see Section 2.4).

Goal Tests and Solutions

After every construction application, one or multiple goal tests are ran over the resulting transient structure to see if it qualifies as a solution. In comprehension, common goal tests include (i) checking whether no more constructions can apply, (ii) checking whether the resulting meaning predicates are integrated in a single network, and (iii) checking whether all input strings have been processed. Goal tests in formulation typically check (i) if there are no more applicable constructions and (ii) if all meaning predicates in the input have been processed. If one of the specified goal tests fail, the search process continues. However, if no more constructions can apply to the transient structure, the FCG engine returns the partial solution.

```
(unique ?y-14)

(mouse ?y-14)

(deep-affection ?x-49 ?y-14)

(linguist ?x-49)

(unique ?x-49)
```

Figure 2.9: Meaning representation of the utterance “The linguist likes the mouse”.

When a solution is found, or a partial solution needs to be used, the final process of the FCG engine starts: rendering. Rendering consists of transforming the final transient structure into the output. In comprehension, the output is a meaning representation that consists of predicates. Thus, the rendering process scans all units in the transient structure, extracts the predicates from the ‘meaning’ feature and draws them as a network. Figure 2.9 illustrates the meaning network that results from the comprehension of the example utterance...
“The linguist likes the mouse”. In formulation, the output is an utterance. Similar to comprehension, all units in the transient structure are scanned, but now the ‘form’ features are extracted. This yields a set of ‘string’ predicates and ordering relations between them. Through a search process, this set of strings is transformed into an utterance that satisfies the ordering relations.

### 2.3.5 Categorial Networks

A recent addition to FCG, that will play a central role in Chapter 6, is the integration of the categorial network. Introduced by Van Eecke (2018, Ch. 4), the categorial network allows to capture a network of emergent grammatical categories that integrates with constructional language processing. Specifically, the categories in the categorial network allow to generalise over feature values that are used in constructions.

#### Untyped Feature Structures

The primary components of FCG, namely transient structures and constructions, effectively boil down to feature structures. The feature structures themselves are essentially made up of features, which are constants, and values, which are constants and logic variables that can again be structured, e.g. in sequences, sets or feature-value pairs (see Section 2.3.4). Crucially, features in FCG are untyped. In other words, there is no underlying type definition system that specifies which features can or cannot be used in a grammar or what type of value each feature can take. This is a major strength of FCG, as it allows the formalism to remain completely open-ended and introduce new features and values at any time. Mainly, it is crucial for being able to learn new constructions on the fly through the meta-layer architecture (see Section 2.2.5). However, the untyped feature structures also make it impossible to specify that one particular feature is related to another feature in an efficient and scalable manner.

#### Motivation: Grammar Learning

The main motivation for incorporating a mechanism that allows to generalise over categories in FCG is the automatic learning of construction grammars. To illustrate this, I provide an example that foreshadows the experiment that I will present in Chapter 6. Part of this experiment consists of a mechanistic model of the cognitive capability of pattern finding (Tomasello, 2003, 2009b). Pattern finding plays a central role in theories of usage-based language acquisition and allows language learners to generalise over reoccurring form-meaning patterns in constructions and thereby capture the compositional structure of language. Specifically, starting with holophrastic constructions, which are idiomatic mappings between complete utterances and their underlying meanings, language learners generalise over these and learn item-based constructions and lexical constructions. Item-based constructions capture similarities from holophrase constructions, both in terms of form
and meaning, and provide open slots for the differences. Lexical constructions, on the other hand, capture those differences and provide arguments for filling those open slots.

Consider a grammar that contains two holophrase constructions: the \textit{what-size-is-the-red-block-cxn} and the \textit{what-size-is-the-blue-block-cxn}\textsuperscript{4}. One could imagine that it is possible to learn the item-based construction \textit{what-size-is-the-?x-block-cxn}\textsuperscript{5}, together with a \textit{red-cxn} and a \textit{blue-cxn}. The item-based construction needs a way to represent the \(?x\) slot and a way to indicate which arguments can be used for that slot. This can be done by adding a unit to the item-based construction, e.g. the ‘?x-slot-unit’, which has a feature-value pair that uniquely identifies the slot, e.g. ‘lex-class: what-size-is-the-x-block-(x)’. To indicate that both the \textit{red-cxn} and the \textit{blue-cxn} are suitable arguments for that slot, the same feature-value pair is added to each lexical construction. During processing, these feature-value pairs can unify (because they are the same) causing either the \textit{red-cxn} or the \textit{blue-cxn} to fill the slot of the item-based construction.

What if the \textit{red-cxn} acts as an argument for another slot, e.g. in the item-based construction \textit{how-many-?x-objects-are-there-cxn}? To solve this, another value needs to be added to the ‘lex-class’ feature of the \textit{red-cxn}, namely ‘lex-class: [what-size-is-the-x-block-(x), how-many-x-objects-are-there-(x)]’. While this is a working solution, it is immediately clear that it is not a scalable one. The ‘lex-class’ feature in both the item-based construction and the lexical construction would be an ever-growing list. This is not only inelegant, it is also inefficient in terms of FCG’s matching and merging algorithm. Consequently, there is a need to represent grammatical categories and their relations in a more efficient manner. As alluded to before, this will take the form of a network of categories.

The Categorial Networks System

A few requirements need to be fulfilled to incorporate a network of categories in FCG. First, the grammatical categories residing in the network need to be coupled to the constructions and the network should be consulted during constructional processing, and in particular during matching and merging. Second, the network needs to be highly dynamic and accommodate for the open-ended nature of FCG. Since new constructions can be introduced at any point in time, this also holds for grammatical categories. Similarly, constructions might become useless or forgotten, requiring the ability to remove categories as well. Third, it should be possible to strengthen or weaken the connection between categories since certain patterns, such as a particular argument for a particular slot, might become more or less entrenched.

The categorial networks system (Van Eecke, 2018, Ch. 4) meets the requirements outlined

\textsuperscript{4}This holophrase construction would map between the utterance “What size is the blue block?” and its underlying meaning representation, e.g. in terms of procedural semantics.

\textsuperscript{5}This item-based construction has an open slot both on the form side and on the meaning side. On the meaning side, this constitutes an unconnected variable in the semantic representation.
above. It consists of two parts. First, it contains a weighted graph data structure containing grammatical categories and their links. These links can either be directed or undirected. This depends on the particular application. Functionalities for adding and removing categories, adding and removing links, updating link weights, etc. are provided. Second, the system provides an adapted version of FCG’s matching and merging algorithm (see Van Eecke (2018, p. 69-70)). Specifically, the algorithm will take the categorial network into account when trying to unify feature values which are constants. Apart from the default case, which checks whether both constants are equal, unification now also succeeds when these constants are connected in the categorial network. How exactly these constants should be connected in the categorial network, i.e. as direct neighbours or via a path, can be specified on a per grammar basis.

Solution for Grammar Learning

Using the categorial network, the example outlined above is solved in an elegant and computationally efficient manner. The solution is illustrated in Figure 2.10. The lexical construction RED-CXN and the item-based construction WHAT-SIZE-IS-THE-?X-BLOCK-CXN are shown at the top of the figure. Both constructions have a ‘lex-class’ feature with a single constant as value. These constants are represented in the categorial network as well. The link between red and WHAT-SIZE-IS-THE-X-BLOCK-(x) in the network indicates that the RED-CXN is a suitable argument for the ?x slot of the WHAT-SIZE-IS-THE-?X-BLOCK-CXN. In the construction application process, shown at the bottom of the figure, the second node shows the resulting transient structure after the RED-CXN has applied. This has created the ‘red-5’ unit in the transient structure. In the following node, the WHAT-SIZE-IS-THE-?X-BLOCK-CXN could apply and the ‘red-5’ unit could indeed fill the slot of the item-based construction. Specifically, the ‘red-5’ unit of the transient structure and the ‘?x-slot-unit’ in the item-based construction unified successfully. As indicated in yellow, unification of these units succeeded due to the link in the categorial network of their respective ‘lex-class’ feature values.

If the RED-CXN is observed as an argument for a slot in another item-based construction, it suffices to specify a ‘lex-class’ feature for the slot in the item-based construction, add its value to the categorial network and create a link from that value to the ‘red’ category. The RED-CXN itself can remain untouched. Similarly, if a new argument of the item-based slot is observed (e.g. blue), the categorial network can be exploited to easily incorporate this observation. It suffices to create a new lexical construction BLUE-CXN that has its own unique ‘lex-class’ feature value and create a link from that feature value to the category that indicates the slot of the item-based construction. The weights on the links in the categorial network can be used to reflect entrenchment, e.g. when a particular argument for a slot is more likely or preferred over another. In Figure 2.10, the weights are set to a default value of 0.5.
Figure 2.10: A schematic representation of a construction application process that uses the categorial network. The item-based construction WHAT-SIZE-IS-THE-?X-BLOCK-CXN can apply after the RED-CXN because the LEX-CLASS feature values RED and WHAT-SIZE-IS-THE-X-BLOCK-(X) are neighbours in the categorial network.
In sum, the categorial network allows to capture emergent generalisations over grammatical categories in FCG without losing the benefits of an open-ended formalism. For other use cases and examples of the categorial networks system, I refer to Van Eecke (2018, Ch. 4).

2.3.6 Neural Heuristics

In Section 2.3.3, I introduced constructional language processing as a state space search problem. The search space involved is combinatorial in nature and, as the grammar increases in size, becomes intractable. This intractability hinders the further development of large-scale construction grammars and their deployment in several domains (van Trijp et al., 2022), such as usage-based linguistics research (e.g. Diessel (2015)), language acquisition models (e.g. Tomasello (2003)) and language technology applications (e.g. Willaert et al. (2020, 2021, 2022); Beuls et al. (2021)). The main challenge in overcoming this intractability problem is to come up with good search heuristics. In FCG, this boils down to determining which transient structure to expand and which construction to apply in order to arrive at a solution.

I discussed several techniques for optimising the search process in Section 2.3.4. Concretely, assigning scores to constructions is typically only used in experiments on the acquisition and emergence of grammar, as these scores reflect the agents’ entrenchment. Assigning these scores by hand is not a scalable solution. Further, the hashing of constructions reduces the number of applicable constructions at every transient structure, but is most effective for morphological and lexical constructions. The search problem for other types of constructions remains. Construction sets are not in line with construction grammar theory and become tedious to manage manually. Finally, priming networks currently offer the best approach for optimising FCG’s search process. They can be learned in a straightforward manner, but they focus only on local information and suffer from scaling issues when using non-local information.

Here, I introduce a novel methodology that allows to learn search heuristics for constructional language processing using an encoder-decoder neural network architecture and its integration in Fluid Construction Grammar. I developed this methodology during my PhD project, but it is not central to the objectives of this dissertation. A paper on this topic has been submitted as Van Eecke et al. (subm). The paper not only introduces the methodology, it also presents a case study on a large-scale grammar that covers more than one million utterances in both the comprehension and production direction. This case study shows that the neural heuristics outperform both FCG’s default depth-first search strategy with backtracking and hashed constructions (see Section 2.3.4) as well as priming networks (Wellens and De Beule, 2010; Wellens, 2011), both in terms of size of the search space and in terms of processing time, and most markedly in the production direction. Importantly, the greatest improvement on both of these metrics and in both directions of processing is achieved
for the inputs that otherwise give rise to the largest search space. An interactive web demonstration of this case study can be found at https://ehai.ai.vub.ac.be/demos/neural-heuristics/. However, a complete description of this case study and the experimental results falls outside the scope of this chapter.

Methodology

The neural heuristics methodology is inspired by recent successes in AI that exploit neural networks for their predictive strength and intelligently combine them with traditional search techniques, e.g. as in AlphaGo (Silver et al., 2016) and planning problems (Takahashi et al., 2019; Wang et al., 2019; Ferber et al., 2020). Applied to constructional language processing, the neural network allows to compute a heuristic value for every transient structure by providing an estimate of the probability that the application of a particular construction to that transient structure will lead to a solution. To compute these probabilities, the network relies on the input, i.e. an utterance in comprehension or a meaning representation in production, and on the sequence of constructions that have already applied in the current branch of the search tree. The transient structures’ heuristic values steer the construction application process via a best-first search.

This methodology is operationalised by two recurrent neural networks (RNNs), one for each direction of processing, that are organised in an encoder-decoder architecture. The encoder network encodes the input before the construction application process starts. During construction application, every transient structure queries the decoder. This receives as input the names of the constructions that have already applied, the encoded input and the encoder’s hidden states. The output is a probability distribution over all constructions in the construction inventory. Then, the transient structure is expanded and the heuristic values for the new transient structures are computed based on these probabilities. Specifically, the heuristic value of the expanded transient structure is the sum of the heuristic value of the previous transient structure and the probability of the applied construction. Processing continues with the transient structure that has the highest heuristic value. This process is schematically illustrated in Figure 2.11.

During constructional processing, the neural network faces the task of predicting the next token (i.e. construction), given a sequential input (i.e. utterance or meaning representation) and a sequence of previous tokens. This corresponds to a single step within a sequence-to-sequence task and motives the choice for the encoder-decoder architecture. However, instead of predicting the whole sequence in one go, the decoder is periodically queried and can be interleaved with FCG’s search and backtracking facilities. RNN-based architecture are typically good at handling this kind of tasks (Sutskever et al., 2014). However, CNN-based and Transformer-based architecture have also been successfully applied to sequence-to-sequence problems (Gehring et al., 2017; Vaswani et al., 2017) and can be readily substituted in the neural heuristics methodology.
CHAPTER 2. BACKGROUND AND TECHNICAL FOUNDATIONS

Figure 2.11: Schematic representation of constructional language processing with neural heuristics. Each transient structure queries the encoder-decoder model to obtain a probability distribution over constructions. This is used to compute the heuristic values (hv) of the expanded transient structures. The search space is explored through best-first search. Figure from Van Eecke et al. (subm).

Training

Training the encoder-decoder architecture requires a dataset of sequential inputs, i.e. utterances in comprehension or meaning representations in production, paired with a sequence of names of constructions that, when applied, leads to a solution. Utterances can be naturally transformed to sequences by tokenizing them. Meaning representations that are by default supported by FCG, namely predicates that share their arguments by linking variables, can be transformed to sequences through reverse Polish notation or postfix notation.

Crucially, in order to obtain training data for this methodology, one is facing the very search problem that will be optimised. Specifically, the available data should first be processed without neural heuristics before training the neural network can start. If this is computationally not feasible, a spiral approach can be followed, where a neural network is trained on part of the data that can be processed within reasonable time and without any heuristics. This first network can then be used to speed up the annotation of more training data, which then allows to train a second neural network. This process can be repeated several times until all data is annotated after which the final heuristic network can be trained. Because of this circular relation between obtaining training data and being able to train the neural network, the neural heuristics methodology is currently most suited for...
2.4 PROCEDURAL SEMANTICS WITH IRL

language technology applications or corpus studies with a relatively fixed grammar and less so for experiments on the emergence of grammar where the construction inventory is in constant flux.

The encoder-decoder network is trained only on the names of the constructions. It does not rely on any other information, such as particular features, the number of units in a construction or the transient structure or how these units are structured. This kind of information might further enhance the predictive capabilities of the neural network. In particular, by using the names of the constructions, the network can predict which construction to apply at any point in time. However, another source of ambiguity is the way in which the constructions should be applied, specifically on which unit(s) of the transient structure. This ambiguity might be resolved by providing the neural network with more information on features, units and their structure, as can be done with priming networks (Wellens and De Beule, 2010; Wellens, 2011).

2.4 Procedural Semantics with IRL

The third and final technical foundation is Incremental Recruitment Language (IRL) (Van den Broeck, 2008; Spranger et al., 2012b), a special-purpose formalism for representing, processing, learning, reasoning over and automatically constructing procedural semantic representations. IRL allows to bridge the gap between agents’ sensorimotor processing and language capabilities in language game experiments. The theoretical foundations of IRL can be found in procedural cognitive semantics (Section 2.4.1). After discussing these foundations, I present the main ideas underlying IRL (Section 2.4.2) and how these are operationalised through IRL’s building blocks (Section 2.4.3). Finally, I show how these building blocks are used in language game experiments for operationalising the process of interpretation as the execution of procedural semantic representations (Section 2.4.4) and the process of conceptualisation as the goal-directed composition of procedural semantic representations (Section 2.4.5). Throughout these sections, I draw parallels between the Incremental Recruitment Language system on the one hand, and constraint programming languages and constraint satisfaction on the other hand. Finally, in Sections 2.4.6 and 2.4.7, I present two extensions to IRL’s composition mechanism, namely chunking and flexible interpretation. The former allows to group conventionalised ways of construing semantics and thereby reduces the search process involved in composition, while the latter allows to hypothesise on how to complete partial semantic networks by combining the composition process with unification of semantic networks.

2.4.1 Procedural Semantics

Procedural semantics was developed quasi simultaneously by Woods et al. (1972) and Winograd (1972). Both researchers developed computational systems that can respond to
natural language questions or commands. The LUNAR system (Woods et al., 1972) allows researchers to query the chemical composition of moon rocks from the Apollo missions through natural language questions. Similarly, the SHRDLU system (Winograd, 1972) can hold coherent conversations with humans about a blocks world. This includes executing commands given in natural language, answering questions about the state of the world or previous commands and asking for clarifications when a command or question was unclear. These capabilities require sophisticated reasoning and grammar components. For both of these systems, the researchers were looking for a way to represent the meaning underlying natural language expressions. They drew parallels between natural and computer programming languages and how these could be processed by humans and machines, respectively. The main assertion of procedural semantics is that the meaning of linguistic expressions can be captured in procedures that can be executed algorithmically. After LUNAR and SHRDLU, procedural semantics was further described by Johnson-Laird (1977) and Woods (1981, 1968). Johnson-Laird (1977) describes procedural semantics through the ‘compile and execute’ metaphor for natural language comprehension and production. Specifically, in comprehension, the ‘compile’ step equates to mentally translating a linguistic expression into a program that expresses its meaning and the ‘execute’ step equates to applying or interpreting this meaning with respect to one’s own perception of the world. In production, a speaker ‘compiles’ a program that it wants the listener to execute in order to achieve a communicative goal.

Over the last decades, a wide range of approaches have been applied to operationalise both the ‘compilation’ of natural language in procedural semantic representations and the ‘execution’ of those representations. Specifically, the compilation step has been tackled by definite clause grammars (Pereira and Warren, 1980) (i.a. Warren and Pereira (1982); Zelle and Mooney (1996) and Kanazawa (2007)), combinatory categorial grammar (CCG) (Steedman, 1987) (i.a. Zettlemoyer and Collins (2005); Kwiatkowksi et al. (2010); Krishnamurthy and Mitchell (2012); Berant et al. (2013); Cai and Yates (2013); Reddy et al. (2014) and Pasupat and Liang (2015)), Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) (i.a. McFetridge et al. (1996) and Frank et al. (2007)), dependency parsing (i.a. Andreas et al. (2016b,a)), context-free grammars (i.a. Wong and Mooney (2007) and Huang et al. (2008)) or recurrent neural networks (i.a. Andreas et al. (2016a); Dong and Lapata (2016); Zhong et al. (2017) and Cheng et al. (2019)). Finally, given that Fluid Construction Grammar (Section 2.3) does not impose a particular kind of meaning representation, it is also well suited for mapping between natural language utterances and procedural semantic representations. This has been applied by, among others, Spranger (2016); Bleys (2016); Pauw and Hilferty (2016) and Nevens et al. (2019a).

The result of the ‘compilation’ process constitutes the procedural semantic representation. Therefore, it should not only be straightforwardly executable, it should also be expressive enough to capture natural language utterances and, ideally, be able to capture the compositional and non-compositional aspects of natural languages. The latter facilitates jointly
learning over linguistic and semantic structures through generalisation processes. Three different approaches can be identified for operationalising the ‘execution’ step. As a first approach, natural language utterances are mapped to query languages, such as SQL (Zhong et al., 2017), FunQL (Cheng et al., 2019) or SPARQL (Yahya et al., 2012). This ties the expressiveness of the semantic representation to the expressiveness of the query language. However, the applicability of these semantic structures is thereby limited to databases. Further, query languages naturally capture interrogative linguistic expressions, but are less suited for other types of expressions. The structure of these query languages is also far removed from the way information is structured in natural language. A second approach makes use of logical forms, often in terms of lambda calculus (see e.g. the work cited above in the context of CCG). Logical forms are inherently more expressive than query languages, they are not restricted to interrogative linguistic expressions and more closely resemble how information is structured in natural language (i.e. through compositionality). The main disadvantage of logical forms is that these are not directly executable. An additional step is required to transform logical forms in executable procedures. A third approach, that includes the Incremental Recruitment Language system, consists of formalisms that are specifically designed to represent and execute procedural semantic representations. Having special-purpose formalisms allows the semantic representations to be tailored towards the task at hand and constructed to better reflect the compositional nature of linguistic expressions. Another major benefit is that special-purpose formalisms allow the procedures to be operationalised in whatever way is best suited for the task. For instance, Andreas et al. (2016b), Johnson et al. (2017b) and Kottur et al. (2018) use fully sub-symbolic procedures in the form of several neural network architectures, Yi et al. (2018); Mao et al. (2019) and Nevens et al. (2019a) use fully symbolic procedures and Manhaeve et al. (2018, 2021) use a hybrid approach that combines symbolic and sub-symbolic procedures, specifically by integrating neural networks in probabilistic logic programs. In Chapter 5 of this dissertation, I develop a hybrid procedural semantics approach that is directly integrated in Incremental Recruitment Language.

2.4.2 Incremental Recruitment Language

The Incremental Recruitment Language (IRL) system was specifically designed in the context the language game paradigm. The development of IRL is motivated by two main observations. A first observation is that models of the emergence and evolution of language not only need to explain lexical and grammatical aspects, but also the underlying semantic aspects. In cognitive semantics, there is a consensus that the ability to use language draws upon general cognitive mechanisms and therefore has to be grounded through embodied sensorimotor experiences (Lakoff, 1987; Croft and Cruse, 2004; Spranger et al., 2012b, p. 154). Empirical research in this field has shown that human speakers conceptualise the meaning they want to express in terms of categories, relations, sets, sequences, perspectives, etc. before formulating this meaning through an utterance (Talmy, 2000; Spranger
et al., 2012b, p. 153). This rich repertoire of conceptualisations has been shown to be language-specific (see e.g. Kay and McDaniel (1978) for colour terms and Levinson (2003) for spatial terms), indicating that these conceptualisations are not innate but acquired and shaped through situated interactions (Spranger et al., 2012b, p. 153). A second observation is that natural languages can express second-order semantics (Dowty et al., 2012). This is illustrated by (Bleys, 2016, p. 18) through the following example:

The adverb ‘very’ in an expression like ‘very big’ modifies the meaning of the adjective ‘big’ instead of being a simple conjunction of predicates for ‘very’ and ‘big’. Furthermore, a ‘big’ predicate can be used in different ways, e.g. to restrict the set of possible referents (e.g. ‘the big ball’), to state a property of an object (e.g. ‘the ball is big’), to make a statement about the predicate itself (e.g. ‘big says something about size’), to compare elements (e.g. ‘this ball is bigger than that one’, etc. (Bleys, 2016, p. 18)

Thus, in order to further advance the research on the emergence and evolution of language, especially in the direction of grammar, a formalism that can capture second-order semantics and is tightly integrated in sensorimotor experience is required.

The Incremental Recruitment Language system allows agents in language game experiments to bridge the gap between their sensorimotor level and language level (see also the semiotic cycle in Section 2.2.2). IRL operationalises procedural cognitive semantics by treating the meanings of utterances as semantic networks that capture the steps that the speaker wants the listener to execute in order to arrive at the shared communicative goal. These networks are made up of primitive cognitive operators and semantic entities. The former capture the basic cognitive capacities of agents, while the latter capture the agents’ conceptual inventories in terms of concepts, categories, prototypes, events, roles, perspectives, etc., and are grounded the agents’ sensorimotor experiences. IRL provides the computational infrastructure for (i) implementing primitive cognitive operators, (ii) representing networks of primitive cognitive operators and semantic entities, (iii) conceptualising semantic networks that satisfy a communicative goal, (iv) interpreting semantic networks in terms of sensorimotor experiences, (v) actively reconstructing semantic networks, (vi) learning new semantic entities and (vii) conventionalising semantic (sub)networks.

Given its origins in the language game paradigm, IRL relies on many of the same ideas and principles. For one, IRL is bidirectional, allowing the same formalism to be used by the speaker, in conceptualisation, and the listener, in interpretation. Further, IRL is completely open-ended, both in terms of the cognitive operators and the semantic entities. Specifically, IRL does not include any operators by default but is a general formalism that allows to specify and operationalise such operators. New semantic entities can be added to the agents’ conceptual inventories on the fly. Finally, as FCG does for linguistic structures, IRL facilitates the self-organisation, selection, level formation and reinforcement learning over semantic structures.
The initial version of IRL is described in Steels (2000a) and Steels and Bleys (2005). A second stable version was presented by Van den Broeck (2007, 2008). More recently, the system was renewed by Spranger et al. (2010, 2012b). Since then, it has been applied in language game experiments on various domains, such as colour (Bleys, 2016), spatial language (Spranger, 2016), quantifiers (Pauw and Hilferty, 2012) and temporal language (Gerasymova and Spranger, 2012).

2.4.3 Building Blocks of IRL

The building blocks of IRL consist of primitive cognitive operators and semantic entities, which are combined into semantic networks or IRL programs. An example IRL program is shown in Figure 2.13. The building blocks of IRL support its two main functionalities: interpreting semantic networks by evaluating IRL programs and conceptualizing semantic networks by composing IRL programs. These are discussed in Sections 2.4.4 and 2.4.5, respectively.

**Primitive Cognitive Operators**

The primitive cognitive operators (or cognitive operators or primitives) capture the basic cognitive functions of an agent. This includes filtering a set, categorizing an object or event, taking an element from a set, taking the union of two sets, performing a spatial transformation, counting a set, etc. In terms of operationalisation, primitives are not restricted to symbolic operations, but they may also call upon neural networks, external web services or APIs of robots or other hardware. A single semantic network can combine cognitive operators that make use of any of these techniques.

Primitives are represented as logic predicates that have a name, e.g. `FILTER-BY-COLOR`, and a list of arguments. Arguments of primitives are variables, e.g. `?target-set`, `?source-set` and `?color`, that are typed. The complete predicate is written as `(FILTER-BY-COLOR ?TARGET-SET ?SOURCE-SET ?COLOR)`. By convention, arguments that are typically (but not always) used as output come before arguments that are typically (but not always) used as input. Primitives can be given scores, which reflect their success in a language game experiment. In order to construct semantic networks, primitives are declaratively combined by reusing variable arguments. On the implementation level, semantic networks are thus unordered sets of predicates.

A primitive cognitive operator represents a multidirectional relationship between its arguments (Van den Broeck, 2007, 2008; Bleys, 2016, p. 18). Depending on the availability of arguments, the cognitive operator uses a different mode of operation. In other words, there are several ways for data to flow in and out of a cognitive operator. Because cognitive operators are represented as predicates with variable arguments, they can also be considered constraints (Van den Broeck, 2007, 2008; Steels and Bleys, 2005). The domains of the arguments depend on their types. In IRL, this is captured by the semantic entities.
A semantic network is thus also a constraint program (Borning, 1981; Sussman and Steele, 1980). Compared to first-order logic, where concepts are typically represented as predicates, these constraints can use concepts as their arguments. This can be thought of as a relational predicate, allowing to consider IRL as a second-order semantics (Steels and Bleys, 2005; Bleys, 2008).

Figure 2.12 illustrates the multidirectionality of a filter-by-color primitive. In this figure, incoming arrows denote that data is available for these arguments (i.e. bound arguments), while outgoing arrows indicate that the primitive will compute values for those arguments (i.e. unbound arguments). Primitive cognitive operators can bind multiple unbound arguments, provide hypotheses by binding the same unbound argument multiple times, or both of these simultaneously. The four operational modes, or cases, of this primitive are used in different situations that occur in language games.

![Figure 2.12](image-url)

Figure 2.12: Illustration of the multidirectionality of a primitive cognitive operator. Depending on the availability of arguments (incoming arrows), the cognitive operator can bind values to different (combinations of) unbound arguments (outgoing arrows).

Figure 2.12a illustrates a first case of the filter-by-color primitive. Specifically, a source set containing three objects, here represented as ‘{o₁, o₂, o₃}’, should be filtered on the basis of a colour, in this case ‘red’. The primitive computes a target set with just one element, represented as ‘{o₂}’. This case is typically used by the listener during interpretation, where the speaker has conveyed information through language, including the word red which provides the binding for ‘?color’, and allows the listener to identify some object in the environment. The argument ‘?source-set’ is derived from the environment, e.g. an already filtered subset. Figure 2.12b illustrates a situation where both ‘?source-set’ and ‘?target-set’ are provided, and the primitive should compute the ‘?color’ that makes such a filtering operation possible. This case typically occurs in a learning situation where the utterance contains a word that the listener does not know. The listener indicates failure to
the speaker, who then provides feedback, e.g. by pointing to the correct object. The listener can now derive the source set from the environment, and the target set from the speaker’s feedback. The filter-by-color primitive now infers the concept that can discriminate the object pointed to by the speaker. This may involve creating a new concept on the fly. In Figure 2.12c only the ‘?source-set {o1, o2, o3}’ is provided to the filter-by-color primitive. In this case, the primitive can compute combinations of colours and resulting target sets. The speaker typically employs this case when it is constructing a semantic network during conceptualisation. Each combination of colour and target set constitutes a possible hypothesis that can be further explored in order to find a semantic network that, for example, discriminates an object. Finally, Figure 2.12d illustrates the case where all arguments are available. In this case, no arguments need to be computed and the primitive will check whether the provided arguments are consistent with each other.

**BIND Special Operator**

Primitive cognitive operators can have any name, except for bind. **bind** is a special operator that allows to introduce semantic entities in the semantic network by binding it to a variable. Similar to primitives, a bind statement can be written as a predicate, e.g. (bind color-category ?color red). In this notation, the semantic entity red, of type color-category, is bound to the variable ?color. As a shorthand, ?color ← red can be used.

Cognitive operators internally make use of the special operator bind to assign values to one or multiple of their unbound arguments. Multiple hypotheses, as in Figure 2.12c, can be created by calling bind multiple times on the same variable(s). When the bind special operator is not called within a case that has unbound arguments, this indicates that the primitive cannot compute values for the unbound arguments and the bound arguments are invalidated. The consistency check (Figure 2.12d) does not call bind but returns a boolean value. Finally, bindings can be given scores, e.g. ?color ← 0.5 red, which can be used to reflect the agent’s certainty or prior success about the semantic entity being bound.

**Semantic Entities**

Semantic entities constitute the data that is being manipulated by the cognitive operators. This includes (i) the conceptual inventory of an agent in terms of categories, prototypes, roles, relations, events, etc., (ii) representations of the agent’s environment, e.g. a world model obtained through grounding, and (iii) intermediate data structures that are passed between cognitive operators, e.g. constructed views on the world model (Spranger et al., 2012b, p. 158).

Semantic entities of the first type, i.e. concepts, are grounded in the agent’s sensorimotor experiences. IRL does not enforce any particular way in which these concepts need to be grounded. On the contrary, it allows to flexibly combine different grounding methods
within the same semantic network. In previous research, a variety of methods have been used for grounding semantic entities (Van den Broeck, 2007; Bleys, 2016, p. 19), e.g. using discrimination trees (Steels, 1996), nearest neighbour classification (Belpaeme and Bleys, 2005b; Spranger, 2013), or radial basis functions (Steels and Belpaeme, 2005).

The same concept can be flexibly used in different cognitive operators. For instance, a \texttt{filter-by-color} primitive can be used to find all objects of a given colour, whereas an \texttt{identify-by-most-similar-color} can be used to find a single object that is closest to a given colour, even if the object is not exactly that colour (example from Spranger et al. (2012b, p. 158)). Both primitives use the same colour concepts in different ways.

Semantic entities are typed. This type information is used in two ways. First, it constrains what semantic entities can be bound to the arguments of primitive cognitive operators during the evaluation of semantic networks (see Section 2.4.4). Specifically, only types of entities that are compatible with, i.e. the same type or a subtype of, the primitive’s argument can be bound. Second, it constrains the way in which the cognitive operators can be composed (see Section 2.4.5).

2.4.4 Evaluating Semantic Networks

Given that primitive cognitive operators are similar to constraints, the evaluation of an IRL program corresponds to a constraint satisfaction problem (Van den Broeck, 2008) that is solved through search with backtracking. Crucially, this evaluation process is completely data-driven. The order in which the primitives are executed cannot be derived from the structure of the semantic network but depends on the availability of data, and on how the data is shared between cognitive operators (Spranger et al., 2012b, p. 162). In turn, the availability of data depends on the particular situation within a language game experiment. Three different situations in which semantic networks are evaluated will be discussed later on in this section. Before that, I describe the search process involved in the evaluation of IRL programs via the four components of a state-space search problem as defined by Russell and Norvig (2009, p. 66), similar to constructional language processing in FCG (Section 2.3.3).

- **The state representation** is the set of bindings of the variables in the semantic network up to the current point in evaluation. These bindings hold the information on how the semantic network fits within the world model and the conceptual inventory of the agent evaluating the network.

- **The initial state** is the set of initial bindings that can be collected from the semantic network. Specifically, the \texttt{BIND} statements are extracted from the provided network, the semantic entities specified in those statements are retrieved in the agent’s conceptual inventory and those entities are bound to the variables in the \texttt{BIND} statements. All other variables in the network are marked as unbound. The initial state
Figure 2.13: Example semantic network, or IRL program, containing four primitive cognitive operators: get-context, filter-by-shape, filter-by-size and unique-entity. Semantic entities circle and big are introduced in the network through bind statements. Bind statements and primitives are represented as predicates that are declaratively combined by reusing variables.

is illustrated in the leftmost box in Figure 2.14, showing the evaluation process of the semantic network of Figure 2.13.

- The operators are the primitives. Specifically, the evaluation engine cycles through the primitives of the semantic network that have not yet been executed. For each of these, the evaluation engine checks the primitive’s pre-conditions to see if it is applicable. A primitive is applicable when elements from the current set of bindings (i.e. current state) can be used as bound arguments in a mode of operation of that primitive. This is checked by matching the types of the values in the bindings to the type specifications of the primitive’s arguments. When those pre-conditions are met, the primitive is executed. The execution of a primitive can have three possible outcomes: (i) new bindings for one or more of its unbound arguments, (ii) invalidating the bound arguments or (iii) information on whether all arguments are consistent. If the bound arguments are invalidated or all arguments were bound but turn out to be inconsistent, that particular branch of the search process is not explored any further and the evaluation engine can backtrack. If the primitive returns new bindings, these make up the primitive’s post-conditions. A new set of bindings (i.e. a new state) is created by adding the primitive’s post-conditions to the current set of bindings. Separate branches in the search space are created when a primitive provides multiple hypotheses for the same variable, and when multiple primitives are applicable given the same set of bindings.

- The goal test checks if the list of bindings is both complete and consistent (Spranger et al., 2012b, p. 161). This is the case when all variables in the semantic network are bound (complete) and all primitives in the network have been executed (consistent).

The search process faced by the IRL evaluation engine thus consists in finding the evaluation order of the primitive cognitive operators in the semantic network in order to find a
CHAPTER 2. BACKGROUND AND TECHNICAL FOUNDATIONS

complete and consistent set of bindings for all variables in the semantic network.

Figure 2.14: Evaluation process of the semantic network shown in Figure 2.13. From left to right, each node (box) represents a cognitive operator (or primitive) that is executed and the resulting list of bindings. A set of bindings that is complete and consistent is found in the rightmost node.

The complete evaluation process of the semantic network from Figure 2.13 is illustrated in Figure 2.14. Each consecutive state shows the primitive that was executed and the resulting set of bindings. The last state (dark green box) constitutes a set of bindings that is both complete and consistent. There is no search in this evaluation process as already many arguments of the cognitive operators in the semantic network are provided through bind statements.

The evaluation of semantic networks is used during interpretation, conceptualisation and learning (Van den Broeck, 2007, 2008). In the following paragraphs, I briefly illustrate the evaluation of semantic networks in these three processes.

**Interpretation**

A semantic network such as the one illustrated in Figure 2.13 typically occurs in the interpretation process of the listener during a language game experiment. When comprehension is successful, the listener will have a semantic network with several bound arguments, typically via bind statements, and how these arguments are structured. The evaluation of such a network corresponds to interpreting the speaker’s utterance from the viewpoint of the listener’s sensorimotor experience and conceptual inventory in order to arrive at an action that needs to be performed in order to complete the communicative task, e.g. pointing to an object. In technical terms, the semantic entity involved in completing the communicative task can be retrieved via the network’s target variable. This is the only variable in the semantic network that occurs just once. In Figure 2.13, the target variable is ‘?topic’.
Figure 2.15: Example semantic network as is typically found during conceptualisation. Only the variable ‘\(?topic\)’ has an initial binding. It is bound to the object that the speaker wants to discriminate.

**Conceptualisation**

Semantic networks such as the one shown in Figure 2.13 typically do not occur during conceptualisation. Instead, during conceptualisation, the speaker is constructing a semantic network and wants to validate it with respect to its own communicative intention. In such semantic networks, only one particular variable, namely the target variable, is typically bound to a value, namely the topic or referent of the interaction. An example of such a semantic network is shown in Figure 2.15. To check if the constructed network indeed satisfies the communicative goal, i.e. leads to the expected topic, the speaker has to evaluate this semantic network.

Part of the evaluation process of the semantic network from Figure 2.15 is shown in Figure 2.16. Since there are more unbound variables in the network, the evaluation process generates a larger search space. This figure also illustrates that whenever a primitive returns more than one binding for the same variable, the search space splits and new state representations are created for every possible binding of that variable. For example, the **filter-by-shape** primitive returns three different bindings for the variable ‘\(?shape\)-category\)’. For one of those bindings, the search space splits again due to the **filter-by-size** primitive. Some of the branches lead to dead ends, because the bindings are invalidated or inconsistent. Only one branch, namely the one that discriminates the topic as a big circle, leads to a solution (dark green node). Often, IRL programs that are evaluated during conceptualisation have multiple solutions. By default, IRL explores the entire search space and returns all sets of bindings that are both complete and consistent. The sets of bindings can be ordered by summing their scores. It is then up to the conceptualisation process to determine which of these solutions is most useful. This will be described in Section 2.4.5.
CHAPTER 2. BACKGROUND AND TECHNICAL FOUNDATIONS

Evaluating IRL program

Applying IRL PRIMITIVE INVENTORY (5) on the ontology IRL ONTOLOGY (3) evaluation process

0, 1.00: initial statuses

1, 2.00: get-context statuses evaluated

3, 2.00: filter-by-shape statuses evaluated

4, 2.00: filter-by-size statuses evaluated

5, 3.00: unique-entity statuses evaluated

reset

Figure 2.16: Part of the evaluation process of the semantic network shown in Figure 2.15. The only initial binding is the variable '?topic'.

Figure 2.15: The search tree splits at both the filter-by-shape primitive and the filter-by-size primitive.
Learning

Figure 2.17: Example semantic network for learning a semantic entity. The structure of the network and the binding for ‘?shape-category’ is the result of comprehending the utterance, while the ‘?topic’ variable is bound via the speaker’s feedback. A semantic entity can be learned for the single missing variable ‘?size-category’.

Figure 2.17 illustrates a semantic network that can be used in a learning situation. In this specific situation, the listener’s comprehension process resulted in a binding for the variable ‘?shape-category’, while the variable ‘?size-category’ remained unbound. At the end of the interaction, the listener was shown the topic ‘object-11’ by the speaker. By fully exploiting the multidirectionality of cognitive operators, a binding for the open variable ‘?size-category’ can be inferred. The FILTER-BY-SIZE primitive may create a new semantic entity on the fly, based on the agent’s current observation, or re-use one from the agent’s conceptual inventory if it leads to a solution when evaluating the network. If multiple semantic entities lead to a successful evaluation of the semantic network, one of them can be chosen for example based on discrimination or saliency. The evaluation result of the semantic network shown in Figure 2.17 can then be used to learn a new form-meaning mapping, where the meaning constitutes the value found for the variable ‘?size-category’. The evaluation process is not illustrated, as it is very similar to the evaluation processes shown in Figures 2.14 and 2.16.

2.4.5 Composing Semantic Networks

I now turn to the automatic composition of semantic networks. This is necessary for the conceptualisation process in language game experiments. In this process, the speaker is “planning what to say” (Steels and Bleys, 2005). In other words, it tries to construct a semantic network that achieves a particular communicative goal when interpreted by the listener, e.g. discriminating a particular object in the current environment. Given that semantic networks can be seen as constraint programs, conceptualisation involves the composition of such constraint programs. This is a combinatorial search problem, as the constraints in IRL can share their variable arguments in many different ways. IRL’s composer mechanism uses a goal-directed, best-first search strategy to navigate this search
space. During my PhD project, I re-implemented the composition process such that it is
easier to use and more highly configurable through a number of parameters. In what fol-
lows, I describe the default behaviour of the composer and I highlight the aspects that can
be configured.

The composer’s search process interleaves constraint satisfaction, i.e. the evaluation of
the constructed constraint program, with eager and incremental search for expanding the
constructed constraint program (Van den Broeck, 2008). Similar to constructional lan-
guage processing (Section 2.3.3) and the evaluation of semantic networks (Section 2.4.4),
I describe this search process according to the four components of a state-space search
problem (Russell and Norvig, 2009, p. 66):

- **The state representation** is the *constraint program* that has been constructed up
to the current point in the composition process, together with the *target variable*,
a list of *open variables* and a *cost*. The target variable captures the communicative
goal of the interaction. It is bound to the semantic entity that the speaker wants the
listener to infer based on the constraint program that it is constructing. For example,
it can be bound to the object that the speaker wants to draw the listener’s attention
to. The open variables represent sub-goals that need to be fulfilled in order to reach
the main goal. Concretely, these are unconnected variables in the current constraint
program. The cost of a state is a heuristic value that is used to determine the order
in which states are explored.

- **The initial state** is an *empty constraint program*, together with the target variable
as the only open variable.

- **The operators** are the *primitives* that can be used to expand the constraint program.
Specifically, the composer cycles through the available primitives and checks their
*pre-conditions*. These pre-conditions hold when adding the current primitive to the
constraint program contributes to the data-flow of that program. This is true when
the primitive has a mode of operation which allows it to compute one or multiple of
the open variables of the current state. Specifically, type specifications of the open
variables are matched against the type specifications of the primitive’s unbound ar-
guments in that mode of operation. The composer thus incrementally extends the
data-flow of the constraint program backwards, starting from the communicative
goal (i.e. the target variable). When adding a constraint to fulfil that goal, this might
introduce new sub-goals which then need to be fulfilled recursively. The primitive’s
*post-conditions* thus consist in extending the constraint program and updating the
list of open variables. At any point, multiple constraints may fulfil the pre-conditions
of the current state. These are then handled in separate branches of the search space.

- **The goal test** checks if the constructed program fulfils the communicative goal by
evaluating it (see Section 2.4.4). Specifically, in the case of discriminating a topic
object, the goal is reached when there exists a data-flow through the constructed program that allows to infer that topic from the agent’s concept repertoire and environment. During this evaluation, the speaker uses its own concept repertoire and world model. The goal test thus corresponds to a form of re-entrance (Steels, 2003). If the evaluation process fails, this indicates that the constructed program is inconsistent, allowing the composer to prune this branch of the search space and backtrack. However, if the evaluation process is successful and allows to infer the topic, any additional bindings that are generated by the evaluation process are added to the constructed program and this program is returned as a solution of the composition process.

The search process faced by the composer mechanism thus consists in composing primitive cognitive operators into a semantic network such that the constructed network fulfils the communicative intention that the speaker wants to convey.

Figure 2.18 illustrates one branch of the composition process. The initial state (top left) has an empty constraint program and the target variable ‘?topic’ as its only open variable. By adding a constraint, in this case UNIQUE-ENTITY, the following state has an extended constraint program and an updated list of open variables. The target variable is no longer part of the list of open variables, as the constraint was added to fulfil that goal. However, the UNIQUE-ENTITY constraint introduced a new open variable ‘?source-set-17’. The FILTER-BY-SIZE constraint could be added next because it has a case that can bind a variable of type ‘object-set’, which corresponds to the type of the open variable ‘?source-set-17’. This process continues until the solution state (dark green box) on the bottom right of Figure 2.18 is reached. This state is a solution because the constraint satisfaction procedure on the constructed program allows to infer the semantic entity that is bound to the target variable. The constraint satisfaction procedure introduced two additional bindings, namely ‘?shape-category-8’ ← ‘circle’ and ‘?size-category-7’ ← ‘big’, which are added to the solution.

**Search Space**

Following Van den Broeck (2008), the size of the search space of the composer can be formally analysed in terms of *multisets*. A multiset is a modified version of a set which allows for multiple instances for each of its elements. The number of instances of each element is called the element’s multiplicity. For instance, in the multiset \{a, b\}, both a and b have a multiplicity of 1, whereas in the multiset \{a, a, a, b, b\}, the multiplicity of a is 3 and that of b is 2. The cardinality of a multiset is the sum of the multiplicities of all of its elements. The number of multisets of cardinality \(k\) from a set of cardinality \(n\), is called the multiset coefficient. It is equivalent to the number of subsets of cardinality \(k\) from a set of cardinality \(n + k - 1\), i.e. a binomial coefficient. Given this
Figure 2.18: One branch of an example composition process. Each node keeps track of the target variable (‘?topic’), a list of open variables and the constructed constraint program. The cost of a node is the second number between parentheses in the node’s name. Only constraints that fulfil a goal (i.e. connect to an open variable and have matching type specifications) are added. The last node constitutes a solution as the constraint satisfaction process allows to infer the topic with two additional bindings: ‘?shape-category-8’ ← ‘circle’ and ‘?size-category-7’ ← ‘big’.
equivalence, the multiset coefficient can be computed as follows:

\[
\binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!} = \frac{n(n+1)(n+2)\ldots(n+k-1)}{k!}
\]

In terms of the composer, the number of potential combinations of \( k \) primitive cognitive operators from an inventory of \( n \) such operators is thus \( \binom{n}{k} \). Assuming an average arity \( a \), the number of potential links between the arguments of \( k \) primitives is given by \( s(k,a) = \frac{(k-1)a((k-1)a+1)}{2} \). The total number of potential constraint programs of size \( k \) is thus approximated by \( \binom{n}{k} 2^{s(k,a)} \), while the total number of intermediate programs with maximum length \( k \), i.e. the size of the entire search space considered by the composer, is approximated by \( \sum_{i=1}^{k} \binom{n}{i} 2^{s(i,a)} \).

**Configurations**

There are typically many different programs that satisfy a given communicative goal. Therefore, the composer keeps track of its current state and can be asked to generate multiple or all solutions up to a certain (configurable) depth. The best program can be selected based on a number of criteria (Van den Broeck, 2008), such as the cognitive effort or level of ambiguity involved in evaluating the program or the expressibility of a corresponding utterance. This can be configured via a function that computes a score and ranks the solutions. Additional goal tests, apart from satisfying the communicative goal, can also be specified.

The expansion operator as described above is the composer’s default behaviour. However, the composer mechanisms allows to specify other expansion operators or even accepts multiple expansion operators that can be combined. For example, an expansion operator can try to connect existing open variables with matching type specifications in the constructed constraint program instead of adding new constraints that compute them. Next to this, the composer also accepts node tests which can be used to prune unwanted intermediate programs that are generated by the expansion operator(s). These node tests are ran before the expanded program is evaluated. Therefore, they rely only on information from the node itself and the structure of the expanded program. Two default node tests are in place. The first limits the depth of the composer search process. The second avoids nodes with identical constraint programs. Indeed, depending on the arity of the constraints and their cases, the same constraint program could be constructed via different paths through the search space. This second node test avoids re-evaluating these constraint programs.

**Cost Function**

The cost of a state is used to determine the order in which nodes are processed. By default, states encountered early in the search space \( (d) \), that have few constraints \( (|p|) \), few open
variables (|o|), few duplicate constraints (|p_d|), and a high score (s) are preferred (Spranger et al., 2012b, p. 166). These elements are combined into a cost function as follows:

\[ c = \frac{d + |o| + |p| + 5|p_d|}{s} \]  

(2.1)

The score of a state is, by default, determined by averaging the scores of the constraints that make up the intermediate program of that state. This cost function results in a best-first search. Other cost functions or functions for scoring constraint programs can be configured.

Both the depth in the search space and the number of primitives are considered in the cost function as these do not necessarily correspond to each other. The reason for this is the chunking mechanism.

### 2.4.6 Chunking

The chunking mechanism in IRL is inspired by the cognitive phenomenon of the same name. Miller (1956) proposed chunks as being the basic organisational unit of human memory that group together meaningful units of information and can be more easily retrieved from memory. In IRL, chunking allows to take (part of) a constraint program and wrap it in a chunk such that it can be used by the composer as if it were an atomic constraint. Given (part of) a program, IRL automatically determines the target variable and open variables of the chunk. A chunk’s open variables are variables that occur just one in the provided (sub)program. The target variable is the open variable that occurs as the first argument of a constraint. Thus, given values for the open variables, the chunk allows to compute a value for the target variable. Figure 2.19 illustrates a chunk that is made up of a constraint network of two constraints. After adding this chunk, the composer can use the name of the chunk, i.e. CHUNK-7, to expand a constraint program in the same way as a regular constraint. Specifically, CHUNK-7 can be used when there is an open variable of type ‘object-set’, and it introduces itself three open variables of types ‘object-set’, ‘size-category’ and ‘shape-category’.

The chunking mechanism is used when a certain constraint program led to a successful communicative interaction. A chunking strategy can then be used to select part of or the complete constraint program, and the resulting chunk can be added to the agent’s memory. Hence, chunking can be implementing using the meta-layer architecture (see Section 2.2.5) through diagnostics that check the outcome of the interaction and repairs that implement chunking strategies. In previous research (i.a. Van den Broeck (2008) and Spranger (2016)), only the most basic chunking strategy was used, namely to use the entire constraint program as a chunk.

The chunking mechanism is motivated both from a computational perspective and from
2.4. PROCEDURAL SEMANTICS WITH IRL

The perspective of language emergence and evolution. From a computational point of view, chunking drastically reduces the complexity of the search space of the composition process. Specifically, the number of possible constraint programs increases exponentially with the number of available constraints. This assumes, however, that all constraints can combine with each other, which is typically not the case (see Van den Broeck (2008) for a formal analysis of the size of the search space). Chunks reduce this complexity by adding multiple constraints in one go, instead of adding one constraint at a time. This avoids a large number of nodes that need to be evaluated at every intermediate depth. Thus, chunking (sub)programs that were used successfully in communicative interactions allows the composition process to jump to a point in the search space that previously proved to be successful (Van den Broeck, 2008). In the context of language emergence and evolution, there is an important relationship between chunking and grammar (Spranger, 2016, p. 38). Specifically, chunks can be used to investigate the different strategies used by agents to conceptualise their environment, how these strategies spread and become conventionalised in the population through communicative interactions. Chunks in IRL accommodate for this by keeping a score, which can be used to reflect this conventionalisation. The relationship to grammar is that the syntactic structure is tightly connected to this underlying conceptualisation strategy. This suggests that certain syntactic markers, such as the use of determiners in English noun phrases or the use of aspect in Russian verbs, are a consequence of the different ways in which speakers of these languages conceptualise their environment (Spranger, 2016, p. 38).

In sum, chunking drastically reduces the cognitive load for the speaker during the conceptualisation process and allows agents to construct ever-more complex semantic structures, e.g. by chunking constraint programs recursively. Additionally, chunks allow to compare conceptualisation strategies and investigate their relationship to grammar.

Figure 2.19: A chunk made up of two constraints. IRL automatically determined the target variable ‘?output-set’ and the open variables ‘?source-set’, ‘?size’ and ‘?shape’. The chunk keeps a score, in this case 0.5, that reflects its entrenchment.
2.4.7 Flexible Interpretation

A final piece of functionality in IRL is the composer’s ability to propose completions of partial constraint programs. In language game experiments, this is beneficial for the robustness of the listener in case of misunderstanding or under-specification of an utterance. In particular, the constraint program that results from comprehending the utterance might have missing \texttt{BIND} statements, missing constraints or missing links. When using \textit{flexible interpretation}, the search process of the composer is similar to the one described in Section 2.4.5. However, the composer does not start from an empty constraint program, but from an incomplete one. Therefore, it searches for a constraint program that simultaneously \textit{matches} with the incomplete program and can infer the communicative goal. Only constraint programs that satisfy both of these conditions are valid solutions. The \textit{matching} operation is performed at every step of the composition search process.

Matching is implemented as a unification-based process between two constraint programs. Spranger et al. (2012b, p. 169) formally describes matching a program \( n \), resulting from comprehension, to the constructed program \( c \) as follows:

- The program \( n \) trivially matches the program \( c \) iff 1) for each bind statement in \( n \) there is an open variable of the same type in \( c \) and 2) every primitive \( p \) of \( n \) is in \( c \).

- The program \( n \) matches the program \( c \) iff there is a function \( f \) from the variables in \( n \) to the variables in \( c \) such that \( f(n) = n' \) trivially matches \( c \), where \( f(n) \) substitutes every variable \( x \) in \( n \) for \( f(x) \).

This matching process can return multiple solutions. Each solution describes a mapping from variables in the comprehended program to variables in the constructed program.

Intuitively, when the composer starts from a partial program, the program that is being constructed must preserve the constraints, the \texttt{BIND} statements and the variable links of the partial program. The \texttt{BIND} statements of the partial program are matched with the open variables of the constructed program via their type information. In order to infer the communicative goal, the composer may add more constraints or variable links to the partial program resulting from comprehension.

2.5 Conclusion

In this chapter, I have introduced the computational tools and techniques that will be used to operationalise subsequent chapters of this dissertation and the broader research context in which these tools and techniques are embedded.

In Section 2.2, I presented the Babel software package as a flexible toolkit that allows to implement and run experiments within the language game paradigm. This experimental paradigm allows to study the emergence and evolution of languages from a cultural per-
2.5. CONCLUSION

spective through multi-agent simulations. In particular, this paradigm tackles the question of how an effective and efficient communication system can emergence in a population of autonomous artificial agents through local, decentralised, and situated communicative interactions. Babel’s ubiquitous meta-layer architecture, which separates routine processing from meta-level processing, allows agents to deal with problem solving, reasoning and learning in a flexible manner across all layers of processing and ensures both robustness and open-endedness.

Most research efforts within the language game paradigm are currently oriented towards the acquisition, emergence and evolution of grammar, which necessitates the use of powerful tools for representing and processing more elaborate semantic and morpho-syntactic structures. The Babel software package includes on the one hand Fluid Construction Grammar (FCG) (Section 2.3) for operationalising processes on the linguistic level and on the other hand Incremental Recruitment Language (IRL) (Section 2.4) for implementing processes on the conceptual level.

Fluid Construction Grammar (FCG) is a special-purpose formalism that provides all the necessary building blocks for operationalising bidirectional constructional language processing. FCG stays close to the basic tenets of construction grammar and aims to be as theory-neutral as possible with respect to the constructional analysis being implemented. In FCG, constructional language processing is treated as a search process, finding the sequence of constructions who’s application leads to a successful comprehension or production process. The integration of the categorial networks system allows to capture emergent generalisations over constructions via a network of grammatical categories. FCG is an open-ended formalism that easily allows to add constructions and grammatical categories on the fly.

Incremental Recruitment Language (IRL) is a special-purpose formalism that provides the necessary building blocks for representing, evaluating, composing, chunking and matching procedural semantic representations. In IRL, the meanings underlying natural language utterances are expressed as constraint networks that capture second-order semantics and can be executed algorithmically over the agent’s sensorimotor experiences and inventories of concepts. The individual constraints represent multidirectional relations between their arguments. This makes the evaluation of constraint networks, i.e. constraint satisfaction, a completely data-driven process, allowing data to flow through the constraint networks in several directions. The flexibility that this brings forward allows agents in language game experiments to use IRL in order to bridge the gap between their sensorimotor processing and language capabilities both for interpretation and conceptualisation. The latter is implemented by the composer mechanism, a highly configurable search process for constructing semantic networks that satisfy a particular communicative goal.

FCG and IRL are tightly integrated in the Babel software package and can be used together seamlessly. Combined, these tools and techniques are well suited to investigate the many
aspects of the acquisition, emergence and evolution of conceptual, morpho-syntactic and semantic structures through situated, communicative interactions.

2.5.1 Contributions

Throughout my PhD project, I made a number of contributions to each of the aforementioned tools. In Babel, the ‘robot-interface’ package, discussed in Section 2.2.4, for the first time incorporates a standardised interface for connecting the Babel software to robotic platforms. This allows researchers to more easily integrate new robotic platforms and operationalise all processes required to set up language game experiments, including the grounding process on the sensorimotor level. This contribution is more extensively discussed in Nevens et al. (2019b).

The novel methodology discussed in Section 2.3.6 allows to learn powerful heuristics for constructional language processing, and in particular FCG, by using an encoder-decoder neural network architecture. This contribution is crucial when it comes to scaling constructionist approaches to language, in particular for situations where a relatively fixed grammar needs to be processed efficiently, such as in corpus studies or language technology applications. In turn, these advances can have further implications in usage-based linguistics or models of language acquisition. A paper discussing the neural heuristics methodology and a case study demonstrating its potential on a large-scale grammar has been submitted as Van Eecke et al. (subm).

My contribution to IRL is on the implementation level. Specifically, I re-implemented both the evaluation process of semantic networks and the composition process of semantic networks. The main goal of these re-implementations is to make them more easily configurable. In particular, given that both of these processes are tackled through search processes, I added a number of configurable setting, e.g. node tests, goal tests, queue regulators, heuristic functions, etc., that allow to steer these search processes in a more modular way. The names of those settings correspond to those used in FCG, which also implements a search process. Additionally, these settings are stored, together with the inventory of primitive cognitive operators, in a new and central ‘primitive inventory’ data structure, reminiscent of FCG’s ‘construction inventory’ data structure. These parallels between IRL and FCG further contribute to the integration of these systems within the Babel software package and make IRL easier to learn and use for both new and more experienced researchers using Babel.
Chapter 3

Data

3.1 Introduction ................................................. 80
3.2 Dataset Requirements ..................................... 80
3.3 The CLEVR Dataset ........................................ 81
  3.3.1 Motivation ........................................... 82
  3.3.2 Images ................................................. 83
  3.3.3 Questions ............................................. 83
  3.3.4 CLEVR CoGenT ....................................... 84
  3.3.5 Statistics ............................................... 85
  3.3.6 Requirement Analysis ................................ 86
3.4 The CLEVR Grammar ......................................... 87
  3.4.1 Motivation ........................................... 88
  3.4.2 Meaning Representation ............................... 88
  3.4.3 Bidirectional Language Processing .................. 92
  3.4.4 Evaluation ............................................ 101
3.5 Conclusion .................................................. 103
  3.5.1 Contributions ......................................... 103

Preface

Part of this chapter, specifically Section 3.4, is based on “Nevens, J., Van Eecke, P., & Beuls, K. (2019). Computational construction grammar for visual question answering. Linguistics Vanguard, 5(1).”
CHAPTER 3. DATA

3.1 Introduction

The two main objectives of this dissertation, as laid out in Section 1.2, are (i) the introduction of novel representations and learning mechanisms that allow autonomous agents to acquire linguistic structures through task-oriented, communicative interactions and that exhibit characteristics found in human languages and (ii) the validation of these novel representations and learning mechanisms through concrete case studies that tackle challenging communicative tasks in concrete environments. In this chapter, I focus on the operationalisation of those case studies and particularly on the task that will be tackled and the dataset that will be used. The use of a standardised, freely accessible benchmark dataset that tackles a well-known and clearly delineated task allows for meaningful comparison with past and future work and makes progress more easily quantifiable. Therefore, I have opted to tackle the task of visual question answering (VQA) on the CLEVR benchmark dataset (Johnson et al., 2017a). In Section 3.2, I first lay out a number of requirements that need to be fulfilled by any given benchmark dataset to be useful for the objectives of this dissertation. Afterwards, in Section 3.3, I introduce the CLEVR dataset, elaborate on its design and discuss how it meets the dataset requirements.

After choosing a benchmark dataset, it can be beneficial to manually construct a solution for it as this can serve as the gold standard for the learning mechanisms introduced in subsequent chapters. I present this solution, namely the CLEVR grammar, in Section 3.4. The CLEVR grammar consists of a computational construction grammar, formalised using Fluid Construction Grammar (FCG), that provides bidirectional mappings between all questions of the CLEVR dataset and their underlying procedural semantic representations, together with an implementation of these procedural semantics through Incremental Recruitment Language (IRL). Together, these components effectively solve the CLEVR benchmark task on the symbolic level. I motivate why the CLEVR grammar was created, demonstrate its language processing capabilities, and discuss its main advantages. Finally, in Section 3.5, I summarize the chapter and reflect on its contributions.

3.2 Dataset Requirements

I specify five requirements that any benchmark dataset needs to meet in order to be useful for the objectives of this dissertation as specified in Section 1.2:

1. **Grounded linguistic expressions.** Languages are not learned in a vacuum. Instead, they are learned through interactions between interlocutors that are situated in a specific environment (Hopper, 1987; Jasperson et al., 1994; Van Eecke, 2018, p. 16) and they draw upon general cognitive mechanisms that are grounded through embodied sensorimotor experiences (Lakoff, 1987; Croft and Cruse, 2004). As laid out in the objectives of this dissertation, the developed methodologies are aimed at autonomous agents. This entails, among others, that these agents have their own
sensors and actuators with which they can observe the environment, act in it and ground their conversation in it. Therefore, the benchmark dataset should not only contain linguistic utterances, but also the environment in which these utterances are used. Such an environment can be realised through simulation, images, video, virtual reality or robot hardware.

2. **Task-oriented.** The benchmark dataset should be able to accommodate a communicative task. Common tasks include reference, (visual) question answering, (visual) dialogue, navigation, etc. The benchmark dataset should therefore not only include linguistic utterances situated in an environment, but it should also be possible to extract or derive the ground-truth solutions to the task.

3. **Scale and similarity.** The benchmark dataset needs to be large enough and contain sufficient variation, both in terms of the linguistic expressions and the environment, to make for a challenging learning problem. At the same time, it should contain linguistic expressions that are sufficiently similar to each other, yet non-identical, as this is a necessary precondition for any kind of generalisation process. This is also fully consistent with the prevailing hypothesis of how children acquire language (Tomasello, 2003; Doumen et al., forthcoming).

4. **Avoid biases.** Ideally, the benchmark dataset should not contain any built-in biases, or avoid them as much as possible, both in terms of the linguistic expressions and the environment. Conditional biases between the linguistic expressions and the ground-truth solutions to the task or between the environment and the solutions should equally be avoided. Learning approaches might exploit any of these biases and learn short-cuts instead of performing actual reasoning.

5. **Semantically annotated (optional).** Learning linguistic structures in terms of construction grammar (see Section 2.3) requires not only linguistic utterances, but also a representation of their meaning since the basic unit of language, namely constructions, consist of form-meaning mappings. While it is perfectly possible to create such a representation from scratch, it can be beneficial to start from an existing semantic annotation.

### 3.3 The CLEVR Dataset

The CLEVR benchmark dataset (Johnson et al., 2017a) was designed to facilitate the development and evaluation of intelligent systems that tackle the task of visual question answering (VQA). First introduced by Antol et al. (2015), the VQA task has become a widely used benchmark in artificial intelligence research as it involves answering natural language questions about images and thereby requires the combination and integration of language processing capabilities, perceptual abilities and multi-modal reasoning.
• Q: Is there an equal number of large things and metal cubes?
• Q: What size is the cylinder that is left of the brown metal things that is in front of the big sphere?
• Q: There is a cylinder of the same size as the brown metal cube; is it made of the same material as the small green ball?
• Q: How many objects are either small cylinders or red things?

Figure 3.1: An example image from the CLEVR dataset (left) and a number of questions (right) exemplifying the various reasoning skills: counting, comparison, attribute identification, spatial relations and logical operations. Image and examples from Johnson et al. (2017a).

Starting with the initial motivation and design choices of the CLEVR dataset in Section 3.3.1, I continue by describing the images (Section 3.3.2) and the natural language questions (Section 3.3.3) of the dataset and how they are generated. Afterwards, I introduce the CoGenT variant of the dataset (Section 3.3.4) and I provide a number of statistics on both the main dataset and the CoGenT variant (Section 3.3.5). Finally, in Section 3.3.6, I motivate the decision for using this benchmark dataset in light of the requirements outlined in the previous section.

3.3.1 Motivation

The CLEVR dataset was specifically created as a diagnostic dataset, enabling detailed analysis of the reasoning processes taking place in systems that tackle the visual question answering task. Since the focus lies on the reasoning aspect, the images are kept simple. They are artificially generated scenes containing a number of 3D shapes of various colours, sizes and materials on a plain background. An example image is shown on the left side of Figure 3.1. Due to their simplicity, these images can be easily processed by off-the-shelf tools and techniques for object detection, instance segmentation or feature extraction. The questions, on the other hand, are designed to test a variety of reasoning abilities such as counting (e.g. “How many spheres are there?”), comparison (e.g. “Are there more spheres than cubes?”), attribute identification (e.g. “What is the colour of the cube?”), spatial relations (e.g. “What colour is the sphere left of the cube?”) and logical relations (e.g. “How many things are either red spheres or blue cubes?”). The right side of Figure 3.1 shows example questions featuring these reasoning skills.

The CLEVR dataset was carefully designed to rule out biases and avoid short-cuts as much as possible. This is a common problem in other VQA datasets, as abundantly shown by Agrawel et al. (2016), Goyal et al. (2017), Manjunatha et al. (2019) and Das et al. (2019). To illustrate this problem, consider the following example from Agrawel et al. (2016). In a
particular dataset, the question “What covers the ground?” will almost always be accompanied by an image of a snowy landscape. This allows the VQA system to learn a strong association between this question and the answer “snow.” This association will be so strong that the answer “snow” will be provided with any input image, without even considering the image and without any reasoning taking place. In generating the CLEVR dataset, a number of measures were taken to avoid these kinds of issues. Most importantly, contrary to most other VQA datasets (e.g. Antol et al. (2015) and Goyal et al. (2017)), neither the images nor the questions were scraped from the web or crowd-sourced. Instead, the scenes are artificially generated with uniform probabilities, such that, for example, a cube is approximately as often blue, as it is green or yellow or red, etc. The questions are generated based on templates that can be automatically instantiated based on the scene. Rejection sampling is used to minimise question-conditional biases (such as the snowy landscape example above) and each question is checked against the image to avoid generating degenerate questions. In sum, these controlled conditions aim to create a dataset that forces the VQA systems to perform actual reasoning instead of finding and exploiting statistical biases from the input data.

### 3.3.2 Images

The images of the CLEVR dataset contain between three and ten geometrical objects on a plain grey background, generated using Blender (Blender, 2018). The objects are placed in the scene such that they are non-overlapping and not completely occluded by one another. Objects can have different shapes, sizes, colours and materials. An overview of the available concepts is provided in Table 3.1. The images are accompanied by a structured, ground-truth annotation. This contains the attributes of each object, their exact positions on the X-, Y- and Z-axis and the spatial relations between them in terms of four spatial relations: behind, left of, right of and in front of. Two important remarks need to be made with respect to the CLEVR images and their annotation. First, the size of the objects is expressed in absolute terms. An object is either large or small, regardless of whether it is placed near or far from the camera’s point of view. Second, the spatial relations operate by projecting the camera’s viewpoint onto the ground plane. Hence, one object is in front or behind another one if its central point is closer or further along the z-axis. Similar definitions are used for left and right, using the x-axis. While this is an unambiguous definition for a computer, it can sometimes be difficult to interpret for humans. For example, when looking at the scene depicted in Figure 3.1, it can be hard to tell whether the leftmost large yellow cube is left or right of the large cyan cylinder.

### 3.3.3 Questions

The questions of the CLEVR dataset are automatically generated based on question templates. Each question template can express a particular question in a number of gram-
Table 3.1: Overview of concepts used in the CLEVR dataset.

<table>
<thead>
<tr>
<th>Category</th>
<th>Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>sphere, cylinder, cube</td>
</tr>
<tr>
<td>colour</td>
<td>blue, brown, cyan, grey, green, purple, red, yellow</td>
</tr>
<tr>
<td>size</td>
<td>large, small</td>
</tr>
<tr>
<td>material</td>
<td>metal, rubber</td>
</tr>
</tbody>
</table>

matically varied ways. For example, one question template contains both “What colour is the $<Z>$ $<C>$ $<M>$ $<S>$?” and “The $<Z>$ $<C>$ $<M>$ $<S>$ is what colour?”. Here, the ‘$<Z>$’, ‘$<C>$’, ‘$<M>$’ and ‘$<S>$’ are template slots that need to be instantiated by a particular size, colour, material and shape, respectively. These instantiations allow for lexical variety since the dataset provides a number of synonyms for shapes, material and sizes. For example, ‘cube’ can also be ‘block’, ‘metal’ can also be ‘metallic’ or ‘shiny’ and ‘large’ is the same as ‘huge’. Instantiations of template slots are sampled randomly from these synonyms. An overview of the available synonyms is provided in Table 3.2. Template slots for sizes, colours and materials can be left out, resulting in many different utterances that can be generated using the same template. For example, the template “What colour is the $<Z>$ $<C>$ $<M>$ $<S>$?” can be instantiated as “What colour is the large rubber cube?”, “What colour is the metal thing?”, “What colour is the ball?”, etc. Across all question templates, fully instantiated questions can contain anywhere between 5 and 45 words, and each question template provides on average four grammatically different forms. The linguistic structure of the question templates features noun phrases, prepositional phrases, anaphora, conjunction and subordination. While these templates often do not reflect actual language use, and some of the larger ones become hardly comprehensible even for humans, they do follow patterns that are consistent with typical English interrogative structures. In total, the CLEVR dataset contains 90 question templates.

In order to properly analyse potential reasoning errors, the question templates not only contain the linguistic form of the question, but also the ground-truth reasoning steps that are required to answer that specific type of question. These reasoning steps are expressed in terms of a library of reasoning functions. Examples of these reasoning functions include scene, for retrieving all objects in the present scene, query_colour for retrieving the colour of a particular object, count for counting the number of objects in a set, etc. An overview of the function catalogue designed by Johnson et al. (2017a), together with examples of how to combine them into reasoning programs, is provided in Figure 3.2.

3.3.4 CLEVR CoGenT

Apart from the main dataset, Johnson et al. (2017a) also provide the CLEVR Compositional Generalisation Test (CoGenT). The goal of the CoGenT dataset is to test compositional gen-
3.3. THE CLEVR DATASET

Table 3.2: Overview of terms and their synonyms available in the CLEVR dataset. Object properties that are used in the CLEVR dataset but missing from this table, such as “cylinder”, “behind”, etc., do not have synonyms.

<table>
<thead>
<tr>
<th>Term</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>thing</td>
<td>object</td>
</tr>
<tr>
<td>sphere</td>
<td>ball</td>
</tr>
<tr>
<td>cube</td>
<td>block</td>
</tr>
<tr>
<td>large</td>
<td>big</td>
</tr>
<tr>
<td>small</td>
<td>tiny</td>
</tr>
<tr>
<td>metal</td>
<td>metallic, shiny</td>
</tr>
<tr>
<td>rubber</td>
<td>matte</td>
</tr>
<tr>
<td>left</td>
<td>left of, to the left of, on the left side of</td>
</tr>
<tr>
<td>right</td>
<td>right of, to the right of, on the right side of</td>
</tr>
<tr>
<td>front</td>
<td>in front of</td>
</tr>
</tbody>
</table>

eralisation abilities of VQA systems. Concretely, this dataset tests whether an intelligent system has truly learned the concepts that are present in the dataset instead of memorising their co-occurrences. This is done by generating images and questions in two experimental conditions. In condition A, cubes can be grey, blue, brown or yellow, cylinders are red, green, purple or cyan and spheres can have of any these colours. In condition B, the colour options for cubes and cylinders are switched and those for spheres remain the same. Apart from these constraints, the images and the questions are generated in the same way as described above. To test the compositional generalisation abilities of a particular VQA system, it should be trained on condition A and evaluated on condition B. Hence, the VQA system will observe combinations of concepts during evaluation that were not observed during training. If the system has really captured the underlying concepts, such as cube, sphere, cylinder, blue, cyan, etc., it should have no issues in transitioning from condition A to condition B.

3.3.5 Statistics

Statistics on the CLEVR dataset are provided in Table 3.3. Across all splits, the CLEVR dataset contains 100,000 images and roughly 1 million questions. Across the three splits, over 85% of the questions are unique. Approximately 12% of the questions from both the validation split and the test split already occur in the training split. Finally, the ground-truth annotations and the ground-truth answers are made available for the training split and the validation split, but not for the test split.

Statistics on the CLEVR CoGenT dataset are provided in Table 3.4. In this table, a distinction is made between experimental conditions A and B. The data splits are nearly identical in size compared to the main dataset. Information on the number of unique questions or
CHAPTER 3. DATA

Figure 3.2: Overview of the CLEVR function catalogue (right) and two examples of how these functions are combined into programs (left). For both programs, the associated question is provided. Figure from Johnson et al. (2017a).

Table 3.3: Statistics on the CLEVR dataset. Table from Johnson et al. (2017a).

<table>
<thead>
<tr>
<th>Split</th>
<th># Images</th>
<th># Questions</th>
<th># Unique Questions</th>
<th>Overlap with train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>100,000</td>
<td>999,986</td>
<td>853,554</td>
<td>-</td>
</tr>
<tr>
<td>Train</td>
<td>70,000</td>
<td>699,968</td>
<td>608,607</td>
<td>-</td>
</tr>
<tr>
<td>Val</td>
<td>15,000</td>
<td>149,991</td>
<td>140,448</td>
<td>17,338</td>
</tr>
<tr>
<td>Test</td>
<td>15,000</td>
<td>149,988</td>
<td>140,353</td>
<td>17,335</td>
</tr>
</tbody>
</table>

the overlap with the training set is not reported by Johnson et al. (2017a). Similar to the main dataset, ground-truth annotations and answers are made available for the training split and the validation set, across experimental conditions, but not for the test split.

3.3.6 Requirement Analysis

The CLEVR dataset fits the requirements outlined in Section 3.2. Specifically, it provides a large corpus of sufficiently similar, but non-identical, compositional utterances due to its template-based question generation process. The utterances are grounded in a specific environment that is realised through artificially generated images. For both the images and
3.4. THE CLEVR GRAMMAR

Table 3.4: Statistics on CLEVR CoGenT dataset.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Split</th>
<th># Images</th>
<th># Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Train</td>
<td>70,000</td>
<td>699,960</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>15,000</td>
<td>150,000</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>15,000</td>
<td>149,980</td>
</tr>
<tr>
<td>B</td>
<td>Val</td>
<td>15,000</td>
<td>149,991</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>15,000</td>
<td>149,992</td>
</tr>
</tbody>
</table>

the utterances, measures are taken to avoid biases as much as possible. Even more so, the CoGenT variation offers an additional instrument to check whether a particular VQA system is prone to bias and is able to generalise. The dataset accommodates a communicative task by design, as it was created for visual question answering, but can also accommodate others tasks such as referential tasks (as in Liu et al. (2019b)) or dialogues (as in Das et al. (2019)). Finally, both the questions and the images are accompanied by ground-truth annotations. For the images, this specifies the properties of the objects, their exact location and the spatial relations between them. For the questions, the annotation details all reasoning steps that are required to answer them, which immediately corresponds to a kind of (procedural) semantic representation.

3.4 The CLEVR Grammar

In this section, I present the CLEVR grammar: a symbolic system that effectively solves the CLEVR benchmark task. This system consists of two components: (i) an FCG grammar that captures bidirectional mappings between the questions and their underlying procedural semantic representations and (ii) an implementation of the procedural semantic representations through the Incremental Recruitment Language (IRL) system that allows to compute the answers to the questions when executed on the symbolic annotation of the CLEVR images. I refer to these components collectively as the CLEVR grammar. Importantly, the CLEVR grammar is not learned but designed by hand as its goal is to serve as the gold standard or as scaffolding for the learning mechanisms that will be introduced in subsequent chapters of this dissertation.

In what follows, I motivate the design of the CLEVR grammar (Section 3.4.1), discuss the meaning representation used in the grammar (Section 3.4.2), demonstrate its bidirectional language processing capabilities (Section 3.4.3), and present evaluation results of the CLEVR grammar in terms of coverage and accuracy (Section 3.4.4). An interactive web demonstration of the CLEVR grammar, including examples of the grammar in both the comprehension and production direction and showcasing the symbolic execution of the procedural semantic representations, can be found at https://ehai.ai.vub.ac.be/demos/clevr-
CHAPTER 3. DATA

3.4.1 Motivation

Both computational construction grammar and procedural semantics are good candidates for tackling the CLEVR benchmark task. Computational construction grammar, in particular, offers an elegant and effective way of dealing with the compositional and non-compositional nature of linguistic expressions, both in terms of form and meaning. This is due to the operationalisation of construction grammar’s basic tenets (described in Section 2.3.1), such as the lexicon-grammar continuum and the tight integration of morpho-syntax and semantics. To illustrate, consider a how-many-X-are-Y construction. Using a single construction, it captures all relevant linguistic information that is associated with utterances that instantiate this construction. This includes, among others, the fact that the meaning (filtering on X first and then on Y and counting the result) only depends on the values of X and Y, the fact that X needs to be a plural noun phrase and Y needs to agree with X, and the fact that “many” cannot be replaced by another determiner or adjective, such as “few” or “numerous”. The main benefit of procedural semantics is that it captures the meaning underlying an utterance in a representation that can be executed algorithmically. This allows an agent that is situated in a CLEVR scene to analyse the question and directly obtain the answer by evaluating the resulting meaning representation. It thereby avoids the highly non-trivial step of transforming a semantic analysis of the question into an executable query language.

Apart from motivating the tools and techniques used to operationalise the CLEVR grammar, there are three main reasons for constructing this solution in the first place. First, it shows that it is possible to solve the CLEVR benchmark task using the aforementioned tools, namely FCG and IRL. This is a good indicator that it will also be possible to learn (parts of) the dataset using the same tools, as they have sufficient representational and processing capabilities to accommodate the solution. Second, having a complete solution allows to use some aspect of this solution as scaffolding, particularly when an experiment focuses on learning only part the CLEVR benchmark task. For example, when trying to learn the semantic concepts present in the dataset, the morpho-syntactic structures can be taken from this solution. Third, this solution can be used to evaluate the results of a learning mechanism or serve as the gold standard.

3.4.2 Meaning Representation

The CLEVR grammar maps the questions of the CLEVR dataset to meaning representations in terms of procedural semantics, implemented using IRL. Specifically, the meanings of the questions consist of networks of primitive cognitive operators (or constraint networks), where each such operator (or constraint) captures a multidirectional relationship between its arguments (see Section 2.4.3). The repertoire of cognitive operators used in the
semantic networks is derived from the catalogue of reasoning functions that is provided with the question annotations of the dataset (see Figure 3.2). However, whereas the CLEVR dataset uses reasoning *functions*, specifying a relation from input to output, the IRL primitives are *constraints*. In Table 3.5, I provide a brief description of each of the 14 primitive cognitive operators as used in the CLEVR grammar. These descriptions focus on the mode of operation of these primitives that is mainly used during the VQA task. However, I also indicate the directions of processing in which each primitive is implemented. The notation \( X \ Y \Rightarrow Z \) indicates a mode of operation where ‘\( X \)’ and ‘\( Y \)’ are bound arguments and the primitive tries to compute a value for ‘\( Z \)’.

Using the primitives outlined in Table 3.5, meaning networks can be built by linking them together through the re-use of variable arguments. An example meaning network is given in Figure 3.3. This network is the semantic representation of the question “What material is the red cube?”. It essentially captures the reasoning operations that need to be performed in order to obtain the answer to that question. The evaluation of this semantic network proceeds as follows. First, the \texttt{GET-CONTEXT} predicate retrieves all the objects in the scene and binds this set to the variable \(?\text{CONTEXT}\). Next, this set is filtered using the concepts \texttt{CUBE} and \texttt{RED}, such that only red cubes remain. The \texttt{UNIQUE} predicate checks whether the set of red cubes contains a single element and binds this element to the variable \(?\text{RED-CUBE}\). Finally, the material of the red cube is retrieved using \texttt{QUERY} and bound to \(?\text{TARGET}\). The binding of this variable is the answer to the question.

![Figure 3.3: A semantic network for the question “What material is the red cube?”](image)

Primitive cognitive operators in IRL manipulate semantic entities, which capture concepts, world models or intermediate data structures (see Section 2.4.3). This is not different for the CLEVR grammar. Apart from the repertoire of primitives listed above, an inventory of the concepts and categories that are present in the CLEVR dataset is provided. This allows to resolve the \texttt{BIND} statements for \texttt{CUBE}, \texttt{RED} and \texttt{MATERIAL} in the meaning network from Figure 3.3. The primitives are designed to operate over symbolic world models instead of
Table 3.5: The primitive cognitive operations used in the procedural semantic representations of the CLEVR grammar.

<table>
<thead>
<tr>
<th>Primitive Cognitive Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>exist bool z set</code></td>
<td>Check if the set is empty or not and bind the boolean result to <code>z</code> bool.</td>
</tr>
<tr>
<td><code>count z set</code></td>
<td>Count the number of objects in the <code>z</code> set.</td>
</tr>
<tr>
<td><code>count z1 z set</code></td>
<td>Check whether concepts <code>z1</code> and <code>z</code> of category <code>cat</code> are equal.</td>
</tr>
<tr>
<td><code>filter z set</code></td>
<td>Filter the set of objects using a concept <code>z</code> of the same category, yielding a subset of the input set where all objects possess <code>z</code>. The resulting subset is bound to <code>z</code>.</td>
</tr>
<tr>
<td><code>for get z set</code></td>
<td>Check if the <code>z</code> set is empty or not. Bind the boolean result to <code>z</code> bool.</td>
</tr>
<tr>
<td><code>for get z1 z cat</code></td>
<td>Filter the set of objects that has the same concept as <code>z</code> for category <code>cat</code>.</td>
</tr>
<tr>
<td><code>for get z1 z cat z set</code></td>
<td>Bind the <code>z</code> set of objects that has the same concept as <code>z</code> for category <code>cat</code> to <code>z</code>.</td>
</tr>
<tr>
<td><code>for get z1 z2 cat</code></td>
<td>Check whether concepts <code>z1</code> and <code>z2</code> of category <code>cat</code> are equal.</td>
</tr>
<tr>
<td><code>for get z1 z2 cat z set</code></td>
<td>Filter the set of objects using a concept <code>z</code> of the same category and the concept <code>z</code> for category <code>cat</code>.</td>
</tr>
<tr>
<td><code>for get z1 z2 cat z set</code></td>
<td>Count the number of objects in the <code>z</code> set.</td>
</tr>
<tr>
<td><code>for get z1 z2 cat</code></td>
<td>A very general concept, for obtaining the category <code>cat</code> of the object <code>z</code> and the concept <code>z</code> for category <code>cat</code>.</td>
</tr>
<tr>
<td><code>get context z</code></td>
<td>Retrieve all objects from the current scene and bind them to <code>z</code> context.</td>
</tr>
</tbody>
</table>

CHAPTER 3. DATA
Table 3.5: The primitive cognitive operations used in the procedural semantic representations of the CLEVR grammar.

<table>
<thead>
<tr>
<th>Primitive Cognitive Operator</th>
<th>Description</th>
<th>Modes of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(equal-integer ?bool ?n1 ?n2)</td>
<td>Check if the numbers ?n1 and ?n2 are the same. Bind the boolean result to ?bool.</td>
<td>?n1 ?n2 ⇒ ?bool</td>
</tr>
<tr>
<td>(less-than ?bool ?n1 ?n2)</td>
<td>Check if the number ?n1 is less than the number ?n2. Bind the boolean result to ?bool.</td>
<td>?n1 ?n2 ⇒ ?bool</td>
</tr>
<tr>
<td>(greater-than ?bool ?n1 ?n2)</td>
<td>Check if the number ?n1 is greater than the number ?n2. Bind the boolean result to ?bool.</td>
<td>?n1 ?n2 ⇒ ?bool</td>
</tr>
</tbody>
</table>
directly processing the CLEVR images. This is illustrated in Figure 3.4, showing an image from the dataset on the left side and the symbolic world model used by IRL on the right side. Because of this symbolic input, the cognitive operators can be straightforwardly implemented using arithmetic operations or set operations.

Figure 3.4: An example image from the CLEVR dataset (left) and the corresponding symbolic annotation (right).

### 3.4.3 Bidirectional Language Processing

The CLEVR grammar consists of 170 constructions for mapping between the questions and their underlying meaning representations. Fifty-five of them are morphological and lexical constructions that were automatically generated based on metadata provided with the dataset. The metadata contains the concepts and categories present in the CLEVR data, together with the available synonyms (see Tables 3.1 and 3.2). The remaining 115 constructions capture the grammatical structures of the questions. These were constructed manually based on the question templates as described in Section 3.3.3.

In what follows, I demonstrate the CLEVR grammar in both the comprehension and the production direction. Through these examples, I provide a glimpse at the constructions that are present in the CLEVR grammar. Afterwards, I highlight two design patterns that were beneficial for the development the CLEVR grammar, and which can be used for developing other grammars with FCG. When describing these patterns, I provide a great number of details on the construction application process. However, the comprehension and production examples should already provide the reader with an idea of how the CLEVR grammar works, allowing the section on these design patterns to be skipped when desired. A complete specification of all constructions, together with interactive examples of comprehension and production, can be found at https://ehai.ai.vub.ac.be/demos/clevr-grammar. For a detailed description of constructional language processing using Fluid Construction Grammar, I refer to Section 2.3.

**Comprehension**

I demonstrate the CLEVR grammar in comprehension by mapping the utterance “What material is the red cube?” to its underlying meaning representation which is shown in Fig-
3.4. THE CLEVR GRAMMAR

Figure 3.5 gives a schematic overview of the comprehension process with respect to the meaning representation that is gradually built up after each construction application. The first construction that applies is the cube-morph-cxn. It does not add any meaning to the transient structure, but only morphological features that can be used later on by the cube-lex-cxn. Before the cube-lex-cxn applies, both the material-lex-cxn and the red-lex-cxn could apply. These three lexical constructions add meaning predicates to the transient structure reflecting the meanings of the words “material”, “red” and “cube”, respectively. These meanings are expressed as bind statements, introducing semantic entities in the meaning network. Separating morphological and lexical constructions, e.g. to analyse the word form “cube” by combining the cube-morph-cxn and the cube-lex-cxn, serves to elegantly handle the synonymy of the CLEVR dataset. This design pattern will be explored in depth later on.

After the lexical constructions, the base-nominal-cxn applies. It adds the filter primitive to the meaning network and its last argument is unified with the variable of the bind statement for “cube”. This indicates that some set of objects, identified by the variable ?source-49, needs to be filtered for cubes. The set of objects resulting from this operation will be bound to the variable ?target-115. The nominal-cxn adds a second filter primitive, taking the set ?target-115 as its input. This filter operation is conditioned on “red” through unification between the filter’s last argument and the “red” bind statement. Afterwards, the unique-determined-cxn applies, adding the unique primitive that checks whether the output of the second filter operation yields a set with a single object in it. Based on the arguments of the filter operations, this should be a red cube. If this is the case, the red cube is bound to the variable ?target-object-31. These three construction again constitute a design pattern present in the CLEVR grammar. They are capable of processing variable-length determined noun phrases, such as “the red cube”, “the small metal thing”, “the large rubber purple ball”, etc. A detailed analysis of this design pattern will also be provided later on.

The last two constructions that apply are the what-t-is-cxn and the hop-query-property-cxn. The former matches on the “What material is” from the input utterance, adds the query primitive to the meaning and the last argument of this predicate is unified with the variable in the “material” bind statement. This specifies that the material of something bound to the variable ?source-88 needs to be retrieved. The latter construction ties this query operation together with the determined noun phrase, constructed earlier. Through unification of variables, it becomes clear that it is the material of the referent of the determined noun phrase that should be queried. The construction also introduces the get-context primitive, which retrieves the set of all objects in the scene, and links it to the input of the first filter predicate. This completes the analysis of the utterance “What material is the red cube?” resulting in the fully-connected meaning network shown in Figure 3.3.
Figure 3.5: Overview of the construction application process for comprehending the question “What material is the red cube?” with respect to the meaning each construction contributes to the transient structure. Solid red lines indicate new predicates that are added to the transient structure. Green dashed lines represent links that are made between predicates by unifying their arguments in construction application.
Production

To demonstrate production in the CLEVR grammar, the meaning network shown in Figure 3.3 is provided as input. Figure 3.6 gives a schematic overview of the construction application process with respect to the utterance that is gradually built up. Below each construction, the figure shows part of the utterance that has been constructed. If word order constraints have been imposed on two or more elements, they are written within a single pair of quotation marks. Furthermore, symbols preceded by a question mark, such as ?cube, indicate that some constraints have been imposed, but the exact morphological form is not yet decided on. This happens later on in the construction application process, based on a number of constraints added by other constructions. The lexical constructions red-lex-cxn, cube-lex-cxn and material-lex-cxn apply first as they can match on the bind statements provided in the input. These constructions add either concrete (i.e. “red” and “material”) or underspecified (i.e. ‘?cube’) word forms to the transient structure. The base-nominal-cxn, which applies next, does not add any word forms or word order constraints to the transient structure. The nominal-cxn, however, imposes that “red” and ‘?cube’ must be adjacent. Afterwards, the what-t-is-cxn adds the word forms “what” and “is” and specifies ordering constraints for “what material is”. Similar ordering constraints for the red ?cube are added by the unique-determined-cxn. This construction also imposes that ‘?cube’ must be singular via other features in the transient structure. Finally, the hop-query-property-cxn imposes that “what material is” is adjacent to “the red ?cube” and the cube-morph-cxn applies due to the singular constraints added earlier and provides the concrete morphological instantiation of ‘?cube’, namely “cube”.

The CLEVR grammar features a considerable amount of lexical and syntactic variation, as discussed in Section 3.3. As a result, the same meaning network can be expressed in many different ways. The lexical variation is handled through the separation between morphological and lexical constructions. The syntactic variation, on the other hand, is handled by the free combinations of constructions, i.e. multiple constrictions that can apply to the same meaning network and collaboratively produce an utterance. The syntactic variation can be explored through FCG’s production process, specifically by exploring the entire search space and returning all syntactically different solutions. Leaving the lexical variation out of the equation, the meaning network shown in Figure 3.3 can be mapped to the following utterances:

• “What is the material of the red block?”
• “What material is the red cube?”
• “What material is the red cube made of?”
• “The red block is made of what material?”
• “There is a red cube; what is its material?”
Figure 3.6: Overview of the construction application process in production, starting from the meaning network in Figure 3.3. Below each construction, part of the utterance that has been constructed is shown. Quotation marks are used to indicate which parts of the utterance are subject to word order constraints. Variables are used to indicate that the exact morphological form of some part of the utterance is not yet decided on.

- “There is a red block; what material is it?”

While producing utterances is not necessary for the CLEVR benchmark task, it can be useful for many other applications, such as chatbots or personal assistants. Furthermore, having shown the possibility to generate syntactically different utterances, the CLEVR grammar can also be used as a paraphrasing tool.

**Design Patterns**

The CLEVR grammar features several design patterns that are commonly used when developing grammars in FCG. In what follows, two of these patterns will be explored in depth. This section is recommended for readers who are interested in a detailed description of the construction application process in the CLEVR grammar.

**Morphological and Lexical Constructions**  The first pattern deals with the lexical synonyms of the CLEVR dataset (see Table 3.2). These synonyms can be used interchangeably, regardless of context, and thus all word forms should map to the same meaning. This is dealt with by separating morphological constructions from lexical constructions. The example for ‘cube’ is shown in Figure 3.7. This figure shows how the transient structure changes by applying the constructions in comprehension. Starting from the top left, the
Figure 3.7: CLEVR’s synonymy is handled by separating morphological constructions from lexical constructions. In comprehension, the former can match on a specific word form from the utterance. The latter matches on morphological features added by the former, generalises over different word forms and provides the meaning.
CHAPTER 3. DATA

initial transient structure contains only the root unit, capturing the input in a feature structure. For space reasons, only a subset of form predicates describing the utterance are shown. In comprehension, morphological constructions match on specific word forms observed in the utterance. In this case, the cube-morph-cxn matches on the form “cube” from the root. This construction does not add any meaning features to the transient structure, but only morphological features. In this case, it adds the lexical class noun, the grammatical number singular and the lexical identifier cube to the transient structure, in a separate unit cube-1. Other morphological constructions for synonyms of “cube”, such as the block-morph-cxn, match on a different word form (i.e. “block”), but add the same lexical identifier to the transient structure. This identifier adds sufficient information for the correct lexical construction to apply next, in this case the cube-lex-cxn. As can be seen from the cube-lex-cxn, it matches not only on the lexical identifier, but also captures the grammatical number provided by the morphological construction through the variable ?number. This is because the same lexical construction is used for both singular and plural word forms. Specifically, the cube-lex-cxn can match on information added by the cube-morph-cxn, the block-morph-cxn, but also by the cubes-morph-cxn and the blocks-morph-cxn. All of these have the same meaning representation, namely the bind statement that is shown in the formulation lock of the conditional part of the cube-lex-cxn. Through this pattern, it is possible to generalise over various word forms which have the same meaning. This avoids having to provide separate constructions for each specific word form, with many features duplicated across those constructions.

Variable-length Noun Phrases The second design pattern deals with analysing the variable-length noun phrases of the CLEVR dataset. In CLEVR, referents can be expressed as “the red cube”, “the large red cube”, “the red metal cube”, “the large red metal cube”, etc. Apart from the determiner, all of these are handled by using just two constructions: the base-nominal-cxn and the nominal-cxn. Briefly put, the base-nominal-cxn looks for the head of a noun phrase and turn this into a nominal group. In CLEVR, the head of a noun phrase is always a noun expressing the shape of an object, or ‘thing’ if the shape is not explicitly specified, or ‘object’ which is a synonym for ‘thing’. The nominal-cxn then looks for an adjective and a nominal group that are adjacent and puts them together to create a new, larger nominal group. This construction can apply recursively in order to add a variable number of adjectives in front of the noun. To ensure that adjectives can only be added in front, the nominal-cxn can only match on the largest nominal group in the transient structure. Additionally, the nominal-cxn construction each time adds a filter primitive to the meaning and correctly links it to the existing meaning network.

Figure 3.8 demonstrates how these construction combine to analyse the nominal group “red cube” in comprehending the utterance “What material is the red cube?”. At the top left, the transient structure is shown after all morphological and lexical constructions have applied. The material-1 unit and the red-1 unit are collapsed for space reasons. First,
Figure 3.8: The base-nominal-cxn and the nominal-cxn are responsible for processing CLEVR’s variable length noun phrases. The former looks for a noun, which constitutes the head of the noun phrase, while the latter can apply recursively, repeatedly prepending an adjective to the noun phrase.
CHAPTER 3. DATA

the base-nominal-cxn can apply. It adds a new unit nominal-unit-1 to the transient structure and modifies the cube-1 unit. The nominal-unit-1 has a filter primitive as its meaning. Notice how the last argument of the filter primitive is unified with the ‘args target’ feature of the cube-1 unit. This indicates that some unknown set of objects (?source) needs to be filtered on cubes which results in another unknown set of objects (?target). These arguments, namely ?source and ?target, are added to the ‘args’ feature of the nominal-unit-1 unit. Re-using variables from primitives into features allows other constructions to more easily match on them and thereby create links in the meaning network. Furthermore, the nominal-unit-1 unit takes over the grammatical number of the cube-1 unit and specifies this unit as its subunit. FCG’s visualisation functions take the subunits feature into account to draw these units hierarchically. Importantly, the nominal-unit-1 unit explicitly adds the feature superunits with an empty value to indicate that this unit corresponds to the largest nominal group at this point in processing. To avoid that the cube-1 unit is identified as the largest nominal group, the feature superunits is added to it, pointing to the nominal-unit-1 unit. Finally, the nominal-unit-1 unit specifies its boundaries in the utterance through the features leftmost-unit and rightmost-unit. As larger structures are being built in the transient structure, it is necessary to keep track of the units that correspond to the word forms at the edges of these larger structures, as only word forms can be used in word order constraints. At this point, the cube-1 unit is both the leftmost boundary and the rightmost boundary of the nominal group.

Next, the nominal-cxn applies. The nominal-cxn looks in the transient structure for both an adjective and the largest nominal group. Importantly, the adjective must be adjacent to the leftmost boundary of the nominal group, as specified by the meets constraint in the ?super-nominal-unit of the nominal-cxn. The largest nominal group can be retrieved by matching on a unit with an empty value for the feature superunits. These conditions are satisfied by the nominal-unit-1 unit. The nominal-cxn adds another filter primitive to the meaning. The last argument ?category of the primitive is unified with the ‘args target’ feature of the adjective. Hence, in this specific case, the filter operation will look for red objects. Additionally, the input set ?between of the new filter primitive is unified with the ‘args target’ feature of the matched nominal group. Through this unification, a link is created in the meaning network. The output of the filter operation for cubes will be the input of the filter operation for red things. What is still unknown are the input of the filter operation for cubes and the output of the filter operation for red things. Therefore, these variables are put in the ‘args’ feature of the newly created super-nominal-unit-1 unit. This allows other constructions to match on these ‘args’ features to further expand the meaning network. Apart from the ‘args’ feature, the super-nominal-unit-1 unit also keeps track of its boundaries (i.e. red-1 on the left and cube-1 on the right), takes over the grammatical number of the nominal group it subsumes (i.e. singular) and adds both the red-1 unit and the nominal-unit-1 unit as its subunits, allowing for hierarchical visualisations. Also, the feature superunits is added with an empty value to indicate that this
is now the largest nominal group. Finally, the superunits feature of the nominal-unit-1 is overwritten, pointing to the super-nominal-unit-1 unit, so that it is no longer eligible as largest nominal group. The red-1 unit also keeps this reference to the super-nominal-unit-1 unit. The resulting transient structure is shown on the bottom right of Figure 3.8. In this transient structure, the cube-1 unit is collapsed for space reasons.

The nominal-cxn can apply recursively. Each time, another adjective will be prepended to the nominal group thereby create a larger nominal group, on which the nominal-cxn can again match. In comprehension, the word order constraints are sufficient to analyse these nominal groups correctly. In production, however, these constraints are not available. This is why the superunits feature is added, such that the largest nominal group can be easily identified and the nominal-cxn can only match on that one. By matching on the superunits feature, the word order constraints are added in the merging phase, allowing the nominal groups to be expressed in the correct order.

The superunits feature is also used to turn the nominal group into a determined noun phrase. Similar to the nominal-cxn, the unique-determined-cxn will look for two elements that are adjacent: a determiner and the largest nominal group. The largest nominal group is again found by matching on the superunits feature with an empty value. The same mechanisms are used, i.e. a larger unit is created that takes both the determiner and the nominal group as its subunits, a new primitive (namely unique) is added, certain arguments in the meaning network are unified, while other arguments, which are still unknown, are passed to the larger unit in the ‘args’ feature such that these can be linked later on. Finally, the larger unit gets an empty value for the superunits feature, while this value is overwritten in the smaller units.

A similar recursive pattern is used to handle variable-length prepositional phrases that are used to analyse questions with spatial relations. Also here, only two constructions, namely the base-relate-cxn and the relate-cxn, are needed to handle questions such as “What color is the large cube left of the metal sphere?”, “What color is the large cube left of the metal sphere behind the purple cylinder?”, “What color is the large cube left of the metal sphere behind the purple cylinder right of the large metal thing?”, etc.

### Evaluation

The CLEVR grammar is evaluated in terms of coverage and accuracy. The coverage of the grammar is evaluated by comparing the output of comprehension, i.e. the meaning networks, to the ground-truth annotations of the questions, i.e. the reasoning programs. This essentially measures the performance on the language understanding part of the benchmark task. The accuracy of the grammar is evaluated by computing the answer to every question using the symbolic annotations of the scenes, and comparing against the ground-truth answer. It is important to note that the aim of the CLEVR grammar is not to compete on the benchmark task itself. This would be an unfair comparison as this solution operates
on the ground-truth, symbolic annotations of the scenes instead of processing the actual images. Instead, the CLEVR grammar serves as scaffolding or as the gold standard solution in subsequent chapters of this dissertation.

Coverage

Computing the coverage of the grammar consists of comparing two directed acyclic graphs (DAGs). On the one hand, CLEVR’s reasoning programs (see Section 3.3) can be easily transformed into a DAG. Every node captures a function with its arguments. The root of the DAG is the last function in the program. Each node thus points to the other node(s) that provided its input. On the other hand, meaning networks as returned by FCG’s comprehension process can be transformed into similarly structured DAGs. Concretely, a node is created for every IRL primitive, using the bind statements in the meaning network to provide the arguments, and the variables through which primitives are linked are used to structure the graph. The coverage of the grammar is now computed by simultaneously traversing both graphs, starting from the root, and comparing each node in terms of the function name and its arguments. If these are equal across all nodes, it can be concluded that the graph data structures are equal and thus the meaning network returned by the grammar is identical to the ground-truth reasoning program.

The above procedure was used on the nearly 850,000 annotated question from training split and the validation split of the CLEVR dataset and 100% coverage was achieved. Importantly, none of these questions were during the development of the grammar. The design of the constructions was based purely on the question templates and hand-written instantiations of them. This result shows that the CLEVR grammar effectively solves the language understanding part of the CLEVR benchmark task.

Accuracy

The accuracy of the CLEVR grammar consists of computing the answer to every question by executing the procedural semantic representations with IRL and comparing the result to the ground-truth. Provided to IRL are the inventory of primitives listed in Section 3.4.2, a repertoire of semantic entities covering all concepts and categories present in the CLEVR data, and a symbolic world model that is constructed by reading CLEVR’s symbolic scene annotations from file. Applied to both the training split and the validation split, totalling almost 850,000 questions over 85,000 scenes, the accuracy of the CLEVR grammar is 100%. This result can be explained as the coverage of the grammar was already at 100% and the IRL primitives operate over a symbolic representation of the images, which would correspond to a flawless perception module operating over the images.
3.5 Conclusion

In this chapter, I have introduced the visual question answering task and the CLEVR benchmark dataset that will be used in case studies to validate the novel representations and learning mechanisms presented in subsequent chapters of this dissertation. First, in Section 3.2, I laid out a number of requirements that any benchmark dataset need to fulfil according to the objectives of this dissertation. Specifically, I require a dataset of linguistic expressions of sufficient complexity and sufficient similarity which are uttered in a concrete environment. This environment should allow for the design of a communicative task and the dataset should ideally provide the ground-truth solutions to this task. Dataset biases in the linguistic expressions, the environments or the solutions should be avoided at all cost. The CLEVR dataset and its CoGenT variant (Johnson et al., 2017a), introduced in Section 3.3, fit these requirements.

Afterwards, in Section 3.4, I introduced the CLEVR grammar. This consists of a grammar, formalised using Fluid Construction Grammar, that provides bidirectional mappings between all questions of the CLEVR dataset and their underlying procedural semantic representations, together with an operationalisation of these procedural semantic representations through Incremental Recruitment Language. As shown in Section 3.4, the CLEVR grammar offers both complete coverage and perfect accuracy on the CLEVR benchmark task, when using the symbolic scene annotations. Given its dependency on the symbolic scene annotations, the aim of the CLEVR grammar is not to compete on the benchmark task. However, both the coverage and the accuracy of this symbolic solution allow to confidently use (parts of) it as scaffolding or as the gold standard solution in subsequent chapters of this dissertation that will focus on learning (parts of) this grammar.

3.5.1 Contributions

This chapter contributes to the objectives of this dissertation through the introduction of the CLEVR grammar (C1). Specifically, I have introduced a computational system that has sufficient representational capabilities to cover the benchmark dataset. The representational capabilities are due to the integration of a computational construction grammar and procedural semantic representations. On one hand, computational construction grammar elegantly and effectively captures both the compositional and non-compositional aspects of linguistic expressions, both in terms of form and meaning, due to the operationalisation of the basic tenets of construction grammar. Procedural semantics, on the other hand, captures the meaning of these linguistic expressions as representations that are directly executable, thereby establishing a tight integration between the semantics and the environment and avoiding the highly non-trivial step of transforming linguistic analyses into executable queries. The learning mechanisms introduced in subsequent chapters of this dissertation will now focus on how (parts of) these representations, i.e. constructions, procedural semantic representations and the underlying semantic concepts, can be learned.
Additionally, subsequent chapters can use the CLEVR grammar as scaffolding or as the desired outcome of the learning mechanism.

This symbolic solution to the CLEVR benchmark task combines the main advantages of grammar-based approaches with a level of accuracy that was previously only achieved using deep learning techniques (e.g. Mascharka et al. (2018); Yi et al. (2018)). These advantages are threefold: (i) the representations used during linguistic processing and the processing mechanisms themselves are completely transparent and human-interpretable, (ii) the grammar is completely open-ended, in the sense that it can always be extended by adding new constructions, e.g. when new concepts or interrogative structures needs to be covered, and (iii) no annotated training data was required for building the grammar as it relies on and captures expert linguistic knowledge.

The CLEVR grammar constitutes a first contribution of this dissertation (C1). Concretely, the CLEVR grammar is one of the first large-scale operational computational construction grammars, covering more than one million utterances in both the comprehension and production direction without sacrificing processing efficiency. It does thereby not only contribute to the scaling of computational construction grammars, it also demonstrates the potential of recent advances Fluid Construction Grammar, such as the introduction of a new high-level notation or the hashing of constructions (Van Eecke, 2018). Finally, the CLEVR grammar corroborates the theoretical underpinnings of the field of construction grammar (see Section 2.3.1) in that it captures the compositional and non-compositional aspects of language in a set of constructions that can freely combine to analyse questions in comprehension or produce questions in formulation.
Chapter 4

Learning Concepts through Discrimination

4.1 Introduction .................................................. 106
4.2 Background and Related Work ............................. 108
  4.2.1 Prototype Theory ....................................... 108
  4.2.2 Version Space Learning ................................. 109
  4.2.3 Neural Approaches ..................................... 110
  4.2.4 Bayesian Approaches ................................... 112
  4.2.5 Reinforcement Learning ................................. 113
  4.2.6 Robotics Approaches ................................... 113
  4.2.7 Discrimination-based Approaches ..................... 116
4.3 Methodology .................................................. 118
  4.3.1 Tutor and Learner ....................................... 118
  4.3.2 Interaction Script ....................................... 118
  4.3.3 Environment ............................................. 122
  4.3.4 Concept Representation ................................. 126
  4.3.5 Tutor Behaviour ......................................... 130
4.4 Experimental Setup ......................................... 130
  4.4.1 Main Experiment ......................................... 131
  4.4.2 Generalisation Experiment ............................ 131
  4.4.3 Incremental Learning Experiment ...................... 131
4.5 Experimental Results ....................................... 132
  4.5.1 Main Experiment ......................................... 132
  4.5.2 Generalisation Experiment ............................ 135
4.1 Introduction

Autonomous agents rely on their own sensors for perceiving the environment in which they are situated, reading out streams of continuous sensorimotor data. Yet, in order to reason and communicate about their environment, the agents require a repertoire of concepts that abstracts away from the sensorimotor level. Without this layer of abstraction, communication would happen by directly transmitting numerical observations from one agent to another. Such a system easily leads to errors, for example when the agents observe the world from different perspectives, when the agents make use of different sensors, or when calibrating the agents is difficult because of changing lighting conditions and other external factors.

In this chapter, I propose a novel methodology that allows autonomous agents to represent and learn a repertoire of symbolic concepts that provides an abstraction layer over their continuous-valued observations. I follow the definition of concept learning proposed by Bruner et al. (1956, p. 233), namely “the search for and listing of attributes that can be used to distinguish exemplars from non-exemplars of various categories”. Hence, a concept consists of a combination of attributes, mapped to a symbolic label. For computationally representing concepts and their attributes, I rely on prototype theory (Rosch and Mervis, 1975). In prototype theory, concepts specify a number of attributes that instances of that concept tend to possess. These attributes have prototypical values and a graded degree of belonging to the concept, with some attributes being more central than others. Building on these two theoretical pillars, autonomous agents thus face two learning problems simultaneously in order to obtain a repertoire of grounded, symbolic concepts. First, the agents need to find out which attributes are relevant for each concept. This requires mechanisms for identifying meaningful combinations of attributes from the sensorimotor data streams. Second, the agents must be able to recognise instances of particular concepts and distin-
4.1. INTRODUCTION

guish concepts from each other. This requires mechanisms for determining the semantic similarity between concepts and instances in terms of their attributes, and for abstracting out the prototypical values of the attributes from the observed instances.

The proposed methodology is operationalised through a language game (see Section 2.2.2) and extends earlier work within this paradigm. Specifically, earlier work focussed on learning concepts that were either limited to continuous data on a single feature channel, such as colour (Bleys, 2016) or spatial position (Spranger, 2012, 2016), or non-continuous data on multiple feature channels (Wellens and Loetzsch, 2012; Wellens, 2012). In this chapter, both of these restrictions are lifted at the same time. I set up a language game in a tutor-learner scenario, situated in an environment that is derived from the CLEVR dataset (Johnson et al., 2017a) (see also Section 3.3). Both the tutor and the learner can act as the speaker or the listener in the language game. Through the communicative task, the learner acquires the perceptual concepts that are present in the dataset, such as SMALL, RED, LEFT, METAL, etc. Learning those concepts requires not only to find relevant attribute combinations (e.g. “r”, “g” and “b” for RED), but also their prototypical values (e.g. “r:215”, “g:76” and “b:44” for RED). Both agents make extensive use of the notion of discrimination, i.e. maximally separating one object from the other objects in the scene, which is an often-used mechanism in experiments on the emergence and evolution of language (see e.g. Steels (1997a); Vogt (2002); Pauw and Hilferty (2012); Wellens (2012); Bleys (2016); Spranger (2016)) as it ensures that the concepts are optimally relevant for the communicative task and the environment in which they are learned and used.

Through a number of experiments, I demonstrate that the learner extracts a set of human-interpretable concepts from its sensorimotor observations and uses them for both conceptualisation, i.e. finding a discriminative concept for an object as the speaker, and interpretation, i.e. finding an object that best matches with a particular concept as the listener. Additionally, I show that the proposed methodology allows for a repertoire of concepts that (i) generalises to unseen settings, (ii) adapts to the environment, and (iii) requires few interactions to learn. Finally, I validate the acquired repertoire of grounded concepts by integrating it in a “downstream” communicative task. Specifically, I incorporate the repertoire of grounded concepts as semantic entities in the procedural semantic representations of the CLEVR grammar (see Section 3.4) and tackle the CLEVR benchmark task. This constitutes a system that is end-to-end human-interpretable, ranging from the perception and categorisation of objects to the language processing of the questions and reasoning to find the answer. I evaluate the agent’s performance in terms of its question answering accuracy.

The remainder of this chapter is structured as follows. In Section 4.2, I discuss background literature on prototype theory and I present existing computational approaches for representing and learning grounded concept. Section 4.3 introduces the language game, including the environment in which the agents operate and how they represent and learn concepts. Afterwards, in Section 4.4, I introduce the experiments, each showcasing a desir-
able property of the introduced methodology. The results of the experiments are presented and discussed in Section 4.5. The integration of the repertoire of grounded concepts in the CLEVR grammar is demonstrated in Section 4.6. Finally, Section 4.7 summarizes the chapter and highlights its main contributions.

4.2 Background and Related Work

In this section, I first sketch the theoretical underpinnings of the concept representation used in this chapter, namely prototype theory (Section 4.2.1), after which I present prior computational models on grounded concept learning using a variety of approaches (Section 4.2.2 to 4.2.6). For each of these, advantages and drawbacks are highlighted from the viewpoint of operationalising the discussed approach in an autonomous agent. Lastly, in Section 4.2.7, I discuss the work by Wellens (2012, Ch. 5), which has served as the basis of the language game experiment presented in Section 4.3.

4.2.1 Prototype Theory

There is a large body of work from philosophy, psychology and linguistics on the study of concepts, how they are represented in the brain, the role they play in cognition and how they can be learned. In this section, I focus on prototype theory as originally conceived by Rosch and Mervis (1975) and further developed by Hampton (1979) and Medin and Smith (1981). The main idea in this view is that concepts are represented through a summary representation, i.e. summarizing one’s observations. A prototype consists of a number of features that are usually found in the members of the concept. Some features can be more central to the concept than other features, which is represented by weighing them. The weights of features can be derived by counting observations and by comparing features across concepts. For example, if a feature often appears in one concept and hardly appears in other concepts, it can be weighed more heavily than another feature which frequently occurs in several concepts. Using this representation, observations can be assigned to concepts by computing a similarity score, taking all features and their weights into account. Importantly, a prototype does not represent the ‘single best example’ of a concept, as in ‘the ideal bird’ or ‘the prototypical bird’. Instead, seemingly contradictory features may be listed in a single prototype. This is illustrated by Murphy (2004, p. 43) using dogs as an example. Under the assumption that there are somewhat more short-haired dogs than long-haired dogs and very few dogs with no hair, all three of these features may be listed in the dog prototype using different weights to reflect their occurrence. Through the same reasoning, the representation of continuous features becomes somewhat more problematic. Again exemplified by Murphy (2004, p. 43), it is intractable to list all possible sizes of a particular bird species, as there are tiny differences between all birds of the same species. One solution would be to treat certain features differently than others. For example, the size feature of a particular bird species can be represented by averaging over the observa-
tions, whereas the size feature of the concept of bird, compared to cat or elephant, is not.

As illustrated by the previous examples, several aspects of prototype theory are specified rather vaguely and are left open to be filled in by specific models. For example, do features specify a prototypical value, a range of prototypical values or a boolean value? How exactly should the features in a concept be organised? How are features obtained? How are the weights determined? Do some features need to be treated differently than others? How are concepts related to each other? Moreover, in psychological experiments, exemplar theory has been shown to better explain how people use concepts in their minds (Medin and Schaffer, 1978; Nosofsky, 1992). Exemplar theory is the complete opposite of prototype theory. There, a concept is not represented by a summary representation but instead it is argued that one remembers all observed exemplars of a concept. When needing to categorise an instance, similar exemplars can be retrieved and used to determine the most similar concept, e.g. by a majority vote. This theory comes with its own disadvantages, such as the enormous cognitive capacity that would be required for storing all those exemplars. More recent theoretical work on concept representations, however, considers a spectrum of concept representations with prototype theory and exemplar theory being on the extreme ends. These models typically consider some point in between (i.a. Medin et al. (1984); Anderson (1991); Love et al. (2004); Johansen and Kruschke (2005); Griffiths et al. (2007); Vanpaemel and Storms (2008)) for example using a mixture of prototypes and exemplars for categorisation, or using exemplars in early learning and transitioning to prototypes later on.

For the purposes of this chapter, prototype theory is used as the main theoretical guideline for the concept representation used by the agents in the language game. Prototype theory is preferred over exemplar theory due to the enormous storage capacity and processing capacity that is required for the latter.

### 4.2.2 Version Space Learning

One method for representing and learning concepts is through version spaces (Mitchell, 1982). In this method, a concept is represented as an area in an N-dimensional space. The dimensionality of the space equals the number of attributes of each concept. Additionally, a concept is bounded by two hypotheses. On one side, the concept is bound by the most specific consistent hypothesis. On the other side, it is bound by the most general consistent hypothesis. A hypothesis is considered consistent when it agrees with all observed examples, i.e. when the hypothesis classifies all observed positive examples as being positive and all observed negative examples as being negative. With this representation, the simplest way of learning concepts is through the candidate elimination algorithm (Mitchell, 1982). Provided with both positive and negative training examples, the algorithm loops over them and updates both bounding hypotheses. The most general hypothesis is updated such that
it covers all positive training examples, including as much as possible of the remaining N-dimensional attribute space, but excluding any negative examples. This hypothesis typically captures an over-estimation of the concept in that it is bound only by the negative training examples. Similarly, the most specific hypothesis also covers all positive training examples, but covers as little as possible of the remaining attribute space. This hypothesis typically captures an under-estimation of the concept in that it is bound only by the positive training examples. Updating the boundaries happens incrementally, looking for the minimal specialization for the most general hypothesis and the minimal generalization for the most specific hypothesis. After having observed all training examples, a concept consists of a number of attributes with an allowed range of values somewhere in between the two bounding hypotheses in the N-dimensional space. A novel observations can be assigned to a concept by comparing it against the hypotheses learned by the algorithm.

A major drawback of the candidate elimination algorithm is its inability to handle noisy data. Noisy or mislabelled training examples can incorrectly update one or both of the boundaries and recovering from such errors is often difficult. On the positive side, because of the relatively simple representation and learning algorithm, concepts represented using version spaces are often human-interpretable and transparent. Furthermore, when the boundaries are allowed to be updated after training, the concepts remain adaptive over time.

### 4.2.3 Neural Approaches

More recent approaches to concept learning are dominated by neural network-based techniques. In the following paragraphs, I discuss deep learning approaches, visual-semantic embeddings and neuro-symbolic systems.

**Deep Learning**

Deep learning approaches for concept learning vary strongly in the neural network architecture, the learning regime (e.g. classification or unsupervised learning), the concept representation (e.g. a label in a classifier or a group of latent variables) and the task or domain in which concepts are being learned (e.g. hand-written characters or generated graphics). Among others, this includes work by Wang et al. (2015), Xu et al. (2018), Dolgikh (2018) and Rodriguez et al. (2019). One line of research within the deep learning paradigm takes inspiration from human concept learning and incorporates this in the models, e.g. Higgins et al. (2016) and Shi et al. (2019). Specifically, the human ability to acquire a concept after only one or a few examples is incorporated through one-shot or few-shot learning. Similarly, the ability to use concepts in order to recognize new exemplars is achieved through incremental learning and memory modules. In general, these approaches yield high levels of accuracy on their respective tasks but require huge amounts of training data and/or training time. Additionally, the concepts are represented in a way that is often not human-
interpretable and the set of concepts that needs to be learned is often predefined and fixed over time. While some of the aforementioned approaches tackle one or two of these issues, none of them tackle all together.

**Visual-Semantic Embeddings**

An alternative approach within the deep learning paradigm is to learn concepts through visual-semantic embeddings. Specifically, these models learn an embedding space where visual data, i.e. images or bounding boxes within images, is embedded close to the embeddings of the corresponding semantic labels. Typically, the embedding space is learned from large-scale annotated corpora, as in Kiros et al. (2014); Faghri et al. (2017); Lu et al. (2019); Chen et al. (2020, 2021) and others. Once trained, the task of the model is to obtain a semantic label given visual data as input, or vice versa. Concept representations in these models can be considered as mappings between embeddings of semantic labels and embeddings of images, making both the internal representation and the processing mechanisms non-interpretable by humans. Additionally, the concepts to be learned should all be specified in advance in the training corpus. Hence, the repertoire of concepts cannot be incrementally extended. The resulting visual-semantic embeddings can be integrated in tasks that require both a visual and a linguistic component, such as visual question answering, referring expression comprehension or following navigation instructions.

**Neuro-Symbolic Approaches**

A repertoire of concepts can also be learned through various neuro-symbolic systems that are aimed at tasks which require both a visual and a linguistic component. In these models, concepts are represented as embeddings and learned during the visio-linguistic task. The neural aspect is used to process both the visual and linguistic inputs, while a symbolic component is used to perform reasoning over these processed inputs. This approach is used by Mao et al. (2019); Han et al. (2019); Stammer et al. (2021) and Whitehead et al. (2021) for visual question answering and by Mao et al. (2021) for grounded language acquisition. In the work by Mao et al. (2019) and Han et al. (2019), different types of concepts are mapped to different embedding spaces, e.g. a different embedding space for colours than for shapes. With this approach, concepts can be added incrementally, either by adding a new vector to an existing embedding space or creating a completely new space for a new type of concept. As with the other neural approaches, high levels of accuracy can be achieved, thereby sacrificing the interpretability of the learned concepts because of their representation as embeddings. However, comparatively speaking, the aforementioned models require few training data, using less than 10% of the available training dataset. As shown by Mao et al. (2019); Han et al. (2019) and Whitehead et al. (2021), their models generalize well to unseen combinations of attributes and were tested on multiple domains, ranging from generated images and template-based questions to real-world images and crowd-sourced questions.
4.2.4 Bayesian Approaches

The Bayesian approach to concept learning consists of learning a probabilistic model for a concept on the basis of observations. There are two main aspects to such a model: the prior probability and the likelihood. The prior probability, i.e. one’s belief about possible hypotheses, avoids generating overly specific hypotheses for a concept. On the other hand, the likelihood, i.e. a probabilistic prediction of the observation, avoids overly general hypotheses. For example, given three positive examples of dogs, the hypothesis that the concept of dog refers to all animals of this particular species is most probable because of both the prior probability and the likelihood. Indeed, some highly specific hypothesis, such as ‘all dogs except Lassie’, is unlikely due to its low prior probability. Alternatively, a more general hypothesis, such as ‘all animals’, has a low likelihood due to the observed examples being very similar. Initial work in this line of research by Tenenbaum (1999a,b), has been extended to word learning (Xu and Tenenbaum, 2007), rule-based learning (Goodman et al., 2008) and visual concept learning (Jia et al., 2013), where concepts are learned directly from image data in the latter.

Extending this approach further is the research centred around the Omniglot dataset (Lake et al., 2015). This is a dataset of hand-written characters from 50 different alphabets. Each character is written by 20 different people and stored as both image and pen stroke data. The main challenge consists of a within-alphabet one-shot classification task: given an exemplar character and an alphabet, identify to which character of the alphabet the exemplar corresponds. This task aims to replicate the ability of humans to acquire a new concept with only a single example. Next to this, there are three other tasks designed to test concept learning-related abilities: (i) parsing of exemplars into parts and their relations, (ii) generating new exemplars of a given concept and (iii) generating new concepts of a particular type.

In their work, Lake et al. (2015) introduce Bayesian Program Learning (BPL) to tackle the Omniglot challenge. Here, concepts are represented as probabilistic generative models trained using the pen stroke data. They are built in a compositional way such that complex concepts can be constructed from (parts of) simpler concepts. In this case, the model builds a library of pen strokes. Characters can be generated by combining these pen strokes in many different ways. This approach has many advantages, including the ability to do one-shot learning and a powerful compositional representation of concepts that allows not only to classify instances of concepts but also to generate them. While this model achieves impressive results, learning through pen stroke data offers a limited range of possibilities. Other researchers have tackled the Omniglot challenge, mostly using neural approaches as reported by Lake et al. (2019). Almost all of them have focussed on the one-shot classification task using the image data as input. As a result, the BPL approach remains the state-of-the-art model for all tasks in the Omniglot challenge.
4.2.5 Reinforcement Learning

Concept learning has also been approached from a reinforcement learning perspective. In this perspective, a concept is regarded as an abstraction over an agent’s states or actions. Abstraction over discrete states can be achieved through tile-coding (Sutton, 1996). Recently, however, following advances in the domain of deep reinforcement learning, abstraction over continuous states is also possible, specifically through function approximation (Mnih et al., 2015). Abstraction over actions is commonly achieved through the use of options (Sutton et al., 1999).

One line of research that is particularly relevant for this chapter is the work by Konidaris and colleagues. In their initial work, they map symbols to sets of low-level discrete states (Konidaris et al., 2014). These states were obtained from the continuous environment and discretised by a classifier. A planning problem is then solved using the symbols as operators in the environment. In later work, the set-based representation was replaced by a probability distribution to better capture the uncertainty about the execution of each high-level step (Konidaris et al., 2015, 2018). Similarly, this approach was validated through a planning problem in a continuous state space, where policies for high-level planning problems in a game environment, such as obtain-key or obtain-treasure, could be computed efficiently.

The aforementioned work by Konidaris et al. (2015) has a number of advantages and disadvantages. On the positive side, the symbolic high-level steps can be represented in a human-interpretable way, as the pre- and post-conditions can be easily visualised in the game environment. Additionally, the model can be learned efficiently with relatively few data points. Specifically, 40 iterations of 100 randomly chosen actions were used to extract the high-level steps. However, as is typical in a reinforcement learning setting, the planning steps are learned through experience. Hence, new planning steps must be learned by collecting new experiences specific to this concept and the resulting steps are relatively domain-specific. No experiments are reported that investigate generality, e.g. would jump-left generalize to other game settings, or adaptivity, e.g. does the concept jump-left change when the game physics change.

4.2.6 Robotics Approaches

There is a large body of work within the robotics community that considers various tasks which are similar to what is referred to as concept learning in this chapter. In what follows, I discuss a number of representative approaches that tackle these tasks.

Perceptual Anchoring

The goal of perceptual anchoring is to establish and maintain a link between a symbol and sensor data that refers to the same physical object (Coradeschi and Safronetti, 2003). This link
should remain stable through time and space, e.g. when an object moves through a robot’s view, when it is covered by another object, or when it disappears and later reappears. A perceptual anchoring system consists of two components: the symbol system and the sensor system. The symbol system can manipulate individual symbols referring to objects as a whole, but also predicates capturing properties of the objects. The sensor system can use different representations for objects, e.g. a set of continuous-valued features or a vector in some embedding space. An anchoring system can be implemented in a bottom-up manner, starting from the sensory level, and in a top-down manner, starting from the symbolic level. In the context of perceptual anchoring, the combination of a symbol, a set of predicates and sensor data can be considered as a single concept.

In recent work, a bottom-up perceptual anchoring system was combined with a probabilistic symbolic reasoning system (Persson et al., 2019). This approach allowed to improve the overall anchoring process by predicting, on the symbolic level, the state of objects that are not directly perceived. There are multiple advantages to this approach. First, the authors achieve high accuracy (96.4%) on anchoring objects and maintaining these anchors in dynamic scenes with occlusions, using relatively little training data (5400 scenes, 70% used for training). Additionally, their system is completely open-ended and allows for incremental learning, since the anchor matching function will simply create new anchors when it encounters previously unseen objects. The anchor matching function, in some way a similarity measure, is closely related to the notion of discrimination. The difference being that discrimination also takes the other objects in the context into account. Finally, the representation of a concept can be human-interpretable, depending on the representation of objects in the sensor system and the corresponding symbols and predicates.

Affordance Learning

Affordances focus on the interaction between the perceptual system and the motor system of an autonomous agent. An affordance can be considered as a learned relation between an action in the environment, caused by the motor system, and the effect observed in the environment, captured by the perceptual system (Şahin et al., 2007). Building on this, the agent can learn concepts in terms of affordances. As proposed by Ugur et al. (2011) and further worked out in Ugur and Piater (2015a,b), affordances can be grouped together in so-called effect categories. These effect categories are then mapped to object properties, obtained through clustering, to form a particular concept. For example, the concept BALL is an object with spherical properties that exhibits the roll-effect when pushed. In these models, the authors use concepts learned through their affordances in plan generation and execution tasks, with an agent being capable of planning the necessary actions involving specific objects to reach a given goal state. This approach offers a more action-centric view on the agent’s world, which is complementary to the approach proposed in this chapter. It not only allows an agent to recognize and describe objects in the world in terms of their features, but also correctly act on them. The concepts that are acquired, combining
effect categories with object properties, offer a transparent and interpretable view. The effect categories are expressed in terms of change in visibility, shape and position, and the object properties are stored in a numerical vector with explainable entries, such as features relating to position and shape (Ugur et al., 2011). Additionally, since the concepts are learned through unsupervised exploration, the proposed model is adaptive to the environment. New concepts can be added incrementally through additional exploration and learned concepts can be progressively updated (Ugur and Piater, 2015b). As is typical in robotics, the proposed approach combines learning in simulation with fine-tuning and validation using physical robots. The concepts considered by Ugur et al. (2011) could be acquired after 4,000 simulated interactions. A robot is then used to validate these concepts in several planning problems. Finally, as the agent assesses the object features that are relevant for each effect category, the resulting mappings offer some form of generality, e.g. a ball exhibits the same effect categories regardless of its colour because these features are not found to be relevant.

Symbol Emergence

Similar to Bayesian approaches, probabilistic models are often used for concept learning in robotics. Specifically, concepts are learned through unsupervised online learning algorithms, combining multi-modal data streams through statistical approaches such as Bayesian generative models or latent semantic analysis (Nakamura et al., 2007; Aoki et al., 2016; Taniguchi et al., 2016, 2017). Through the integration of data streams, the acquired concepts constitute mappings between words and objects, as studied by Nakamura et al. (2007) and Aoki et al. (2016), or between words and spatial locations, as studied by Taniguchi et al. (2016, 2017). The latter further used these concepts to aid a mobile robot in generating a map of the environment without any prior information. The statistical methods have the advantage of being able to infer a considerable amount of information from a limited number of observations, and are therefore suitable for use in robotics scenarios. Additionally, they offer interpretability to a certain extent, through a graphical model representation such as a Bayesian network. Finally, the proposed models are adaptive to changes in the environment and offer incremental learning through their online learning algorithms. For a more comprehensive overview on symbol emergence from the viewpoint of robotics, I refer to Taniguchi et al. (2018).

Language Grounding in Robotics

Alomari et al. (2022) present a model for language grounding and acquisition, operationalised using three different robotic platforms. Their model integrates components for (i) learning visual concepts, (ii) grounding these in language through n-grams and (iii) probabilistic grammar induction. Focussing on the visual concepts, these capture object properties, spatial relations and actions. Specifically, a robot is presented with video fragments where RGB-D data is available for every frame in the fragment. Objects can
be recognised and tracked across the frames. Using unsupervised probabilistic methods, specifically a Gaussian mixture model (GMM), specific features that are extracted from each object in the RGB-D data are clustered. Such a cluster represents a concept, e.g. a cluster of HSV values represents a colour concept. Spatial relations are learned in a similar way, by clustering features that are extracted from pairs of objects. For each type of concept, e.g. colours, the features used for clustering are specified in advance, e.g. HSV. When presented with a novel observation, the GMM can determine whether an existing cluster should be updated or a new cluster should be learned. Complex concepts representing robotic actions such as ‘put on top’ or ‘move’ are learned by finding patterns in transitions from one cluster (i.e. object features or spatial relations) to another over multiple frames in the video. The techniques and methodologies used in this work offer a number of desirable properties, similar to those specified in this chapter. Specifically, Alomari et al. (2022) argue for an incremental approach, without the need to specify the number of concepts to be learned in advance. Further, the representation of the concepts is completely transparent and they can be obtained using relatively few data points, as compared to neural approaches.

Among the various approaches to concept learning discussed so far, the approaches from the robotics literature are most closely related to the approach proposed in this chapter, as many of these studies deal with similar issues such as grounding, adaptivity, generality, incremental learning and data-efficient learning.

### 4.2.7 Discrimination-based Approaches

The experiment presented by Wellens (2012, Ch. 5) serves as the basis for this chapter. Wellens uses the language game methodology to study multi-dimensionality and compositionality during lexicon emergence in a population of agents. In his language game, called the compositional guessing game, the speaker tries to draw the attention of the listener to a particular object in a shared scene. Each object in the scene is observed by the agents as a collection of symbolic attributes, e.g. object-1 consists of the attributes \{A-1, A-2, A-3, A-4, A-5\}. The words used by the agents have one or multiple of these same symbols as their meaning (multi-dimensionality). For example, the word “bolima” has the set \{A-1, A-3, A-5\} as its meaning and the word “wabazu” has \{A-2, A-4, A-5\} as its meaning. The agents can use multiple words to describe a particular object (compositionality), e.g. object-1 can be described by saying “bolima wabazu”. At the end of a game, the speaker gives feedback on the outcome of the game and points to the intended object in case of failure, allowing the listener to update its lexicon. This setup leads to a large amount of uncertainty for the agents, as they need to learn what part of the meaning corresponds to what word in the multi-word utterance.

In his work, Wellens proposes two distinct types of strategies for reducing this uncertainty: competitive strategies and adaptive strategies. Both make use the notion of discrimination,
i.e. maximally separating one object from the others, for both conceptualisation (used by
the speaker) and interpretation (used by the listener). In competitive strategies, the agents
explicitly enumerate competing hypotheses, i.e. identical words with a different mean-
ing, each with a different score. Mechanisms are in place to gradually prune this enum-
eration by increasing and decreasing scores. However, this enumeration soon becomes
intractable in environments with many objects or many attributes per object. Adaptive
strategies avoid this enumeration of competing hypotheses. Instead, only a single mean-
ing composed of a set of attributes is kept for each word. Over many games, this meaning is
gradually being shaped based on the feedback provided after each game. How this shaping
is implemented depends on the particular sub-strategy within the adaptive strategy. The
main idea of adaptive strategies is to focus on re-use, allowing agents to use words to refer
to objects even when the associated meanings of the words are not (yet) fully compatible
with the object they want to refer to. Figure 4.1 illustrates the difference between the two
types of strategies.

![Figure 4.1](image)

Figure 4.1: In competitive strategies, competing hypotheses are enumerated in the agent’s
lexicon. Adaptive strategies allow the meaning of words to be shaped gradually by using
a set of attributes. Adding weights to each attribute allows for even more fine-grained
updates.

Within the adaptive strategies, a distinction is made between the baseline adaptive strat-
ey and the weighted adaptive strategy. In the baseline adaptive strategy, the ideas under-
pinning adaptive strategies are implemented in a rather crude way. The agents gradually
shape the meaning of words simply by adding or removing attributes from the set, based
on the feedback after the game. The weighted adaptive strategy offers a more gradual
shaping of the meaning. Here, the meaning is represented as a weighted set of attributes.
Each attribute receives a score, expressing the certainty that the attribute is important for
the word it is linked to. Based on the received feedback, agents can not only add or remove
attributes, but also alter the scores of attributes to reflect changes in certainty. Over time,
the meanings are shaped to capture attribute combinations that are functionally relevant
in the environment, driven by the communicative task and the notion of discrimination.
For more details about the compositional guessing game, the various strategies and exper-
imental results, I refer to Wellens (2012).
In this chapter, concepts are also represented as weighted attribute sets. However, where Wellens (2012) considers multiple, symbolic attributes, I extend this approach to multiple, continuous-valued attributes, thereby introducing the need for more sophisticated representations and processing mechanisms.

4.3 Methodology

The goal of the experiment is for an agent to distil meaningful concepts from streams of continuous sensory data through a series of task-oriented communicative interactions, i.e. a language game (see Section 2.2.2). These interactions are set up in a tutor-learner scenario and take place in an environment that is derived from the CLEVR dataset (Johnson et al., 2017a) (see Section 3.3). Driven by the communicative task and the notion of discrimination, the learner gradually shapes its repertoire of concepts such that it is functional in the environment. In this section, I elaborate on the initial conditions for the tutor and the learner (Section 4.3.1), the interaction script of the language game (Section 4.3.2), the environment in which the agents operate (Section 4.3.3), the concept representation and learning mechanism used by the learner (Section 4.3.4) and the mechanisms used by the tutor (Section 4.3.5).

4.3.1 Tutor and Learner

There are only two agents in this language game: the tutor and the learner. The tutor is an agent with an established repertoire of concepts, while the learner starts the experiment with an empty repertoire. Furthermore, the tutor has access to a high-level symbolic annotation of the scene, while the learner observes the objects in the scene through streams of continuous data. The tutor, in particular, uses the symbolic ground-truth annotations of the scenes that are provided with the CLEVR dataset (see Section 3.3.2). These annotations describe the objects in the scene with all of their properties. Using such a symbolic annotations for the tutor avoids having to manually design a number of concepts in terms of the continuous data streams, which could bias the system. How exactly the agents internally represent these scenes will be explained in greater detail in Section 4.3.3.

4.3.2 Interaction Script

The communicative task in the language game is a reference task. The agents described above are randomly assigned the discourse roles of speaker and listener. Then, the speaker chooses an object, called the topic, and tries to describe it using a single word tied to a concept that discriminates the topic. The task of the listener is to point to the object intended by the speaker. This task is operationalised by following a predefined interaction script according to the semiotic cycle (see Section 2.2.2). Figure 4.2 provides a schematic overview of this script. It consists of the following steps:
4.3. METHODOLOGY

1. **Role Selection (both agents)** The discourse roles of speaker and listener are randomly assigned to the tutor and the learner, in contrast to the experiment described in Nevens et al. (2020) where the tutor always acts as the speaker.

2. **Scene Selection (both agents).** A random scene is selected from the CLEVR dataset.

3. **Feature Extraction (both agents).** Both agents observe the scene. The tutor obtains the high-level symbolic annotation, while the learner extracts streams of continuous data.

4. **Topic Selection (speaker).** The speaker chooses one object from the scene as the topic.

5. **Conceptualisation (speaker).** In conceptualisation, the speaker tries to find a concept that discriminates the topic. Before conceptualisation, however, the symbolic scene annotation is first checked to see whether the topic can indeed be described discriminatively using a single concept. This is not always possible due to the design of the CLEVR dataset. When this is not possible, the speaker samples another object or another scene altogether when all objects were tried unsuccessfully. When the tutor acts as the speaker, it uses the symbolic annotation for conceptualisation. The learner, on the other hand, uses its own repertoire of concepts and the continuous data-streams to find a discriminating concept for the topic. By looking for a concept that is discriminative, the speaker is actively trying to help the listener in solving the communicative task.

6. **Production (speaker).** In production, the speaker utters the word that is associated with the concept chosen in the previous step.

7. **Comprehension (listener).** In comprehension, the listener receives the word uttered by the speaker and checks its repertoire of concepts. If the concept denoted by this word is unknown, the listener indicates failure. This can only occur when the tutor utters a word that has not yet been acquired by the learner.

8. **Interpretation (listener).** If the listener does know the word, it will try to interpret the corresponding concept in the current scene. In other words, the listener will look for the object that best matches the concept and point to it. As in conceptualisation, the tutor uses the symbolic annotation to do this, while the learner makes use of the continuous data-streams.

9. **Feedback & Learning (both agents).** The speaker decides the outcome of the game (success or failure) by checking if the listener pointed to the intended topic. In the case of failure, the speaker provides feedback by pointing to the intended topic. Depending on the assigned discourse roles, learning is possible through the application of the adoption or alignment operators.
Figure 4.2: During a single interaction, both agents observe a scene from the CLEVR dataset. The speaker chooses a topic and produces a word denoting a concept that discriminates this topic. The listener looks up this word in his repertoire. If the word is known, the listener tries to interpret it in the scene. Otherwise, the listener indicates failure. After the interaction, the tutor provides feedback to the learner, allowing it to learn.
Adoption

If the learner acted as the listener and the word uttered by the tutor was unknown, a new word-concept mapping can be learned. At this point, the learner has no way of knowing which attributes are important for the concept. It does know, however, that the tutor used the concept of which the word was uttered to discriminate the topic. Hence, the learner stores an exact copy of the topic with all of its attributes as the initial hypothesis for the concept. Each attribute receives an initial score of 0.5, reflecting the uncertainty that the attribute is relevant for the newly created concept. Attributes scores are bound between 0 and 1.

Alignment

If the learner acted as the listener and it did know the word uttered by the tutor, it can refine its representation of the corresponding concept using the positive example acquired via the tutor’s feedback. This involves shifting the prototypical values towards the positive example and updating the certainty scores of the attributes of the concept. These mechanisms are discussed in Section 4.3.4. Alternatively, if the learner acted as the speaker, the only learning opportunity occurs when the interaction was successful. Indeed, a successful interaction indicates that the topic is a positive example for the concept that the learner used during conceptualisation. This allows the learner to update this concept, both in terms of the prototypical values and the certainty scores. In all other cases, the topic is a negative example and no mechanisms are put in place to update the concept on the basis of negative examples.

Metrics

To evaluate the learner agent, both communicative success and concept repertoire size are measured. Communicative success indicates whether or not the interaction was successful. In other words, it tells if the learner could successfully use a concept in conceptualisation or interpretation. Also, it allows to monitor the number of interactions required to reach a certain level of communicative success, indicating the speed at which the agent is learning. A distinction is made between overall communicative success and communicative success given conceptualisation. The latter metric, introduced by Loetzschr (2015, p. 97), measures the success of the interaction taking into account whether or not the learner could find a discriminating concept for the topic when acting as the speaker. If the learner failed at conceptualisation, the success of the previous interaction is recorded. In all other cases, i.e. if the learner could conceptualise or when it is not acting as the speaker, the success of the current interaction is recorded. The ‘communicative success given discriminative success’-metric was not part of the experiment discussed in Nevens et al. (2020). The second metric, namely the learner’s concept repertoire size, allows to monitor the number of interactions that are required for learning all concepts known by the tutor.
4.3.3 Environment

The agents’ environment is based on the CLEVR dataset (Johnson et al., 2017a). As discussed in Section 3.3, this dataset was specifically designed to avoid biases as much as possible. In practice, this means that across the scenes, there will be as many blue objects as red objects, as many cubes as cylinders, etc., making it adequate for concept learning experiments. There are 19 concepts to be learned in total. These are summarised in Table 4.1. Note that the synonymy from the CLEVR dataset has been removed (compare to Table 3.2). For example, the learner agent only has to acquire a sphere concept and there is no synonymous ball concept. There is thus a one-to-one correspondence between words and concepts in this experiment. In other words, the agent does not face the issues associated with the introduction of synonyms or homonyms into the lexicon. However, overcoming these issues has been extensively studied using the language game paradigm, particularly in the Naming Game (Steels, 1995) and various guessing games (see Section 2.2.3).

Table 4.1: All 19 concepts in the experimental environment.

<table>
<thead>
<tr>
<th>Shapes</th>
<th>Colours</th>
<th>Sizes</th>
<th>Materials</th>
<th>Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBE</td>
<td>BLUE</td>
<td>LARGE</td>
<td>METAL</td>
<td>BEHIND</td>
</tr>
<tr>
<td>CYLINDER</td>
<td>BROWN</td>
<td>SMALL</td>
<td>RUBBER</td>
<td>FRONT</td>
</tr>
<tr>
<td>SPHERE</td>
<td>CYAN</td>
<td></td>
<td></td>
<td>LEFT</td>
</tr>
<tr>
<td></td>
<td>GREY</td>
<td></td>
<td></td>
<td>RIGHT</td>
</tr>
<tr>
<td></td>
<td>GREEN</td>
<td></td>
<td></td>
<td>LEFT</td>
</tr>
<tr>
<td></td>
<td>PURPLE</td>
<td></td>
<td></td>
<td>RIGHT</td>
</tr>
<tr>
<td></td>
<td>RED</td>
<td></td>
<td></td>
<td>RIGHT</td>
</tr>
<tr>
<td></td>
<td>YELLOW</td>
<td></td>
<td></td>
<td>RIGHT</td>
</tr>
</tbody>
</table>

The learner agent observes the environment through streams of continuous-valued sensor data. To achieve this, the symbolically annotated CLEVR scenes, as illustrated in Figure 4.3, need to be transformed into numerical data. Two ways of making this transformation are considered. First, I discuss manually designed rules and procedures to transform the symbolic annotation into numerical data. For the second method, I use a Mask R-CNN model (Yi et al., 2018) to detect and segment the objects directly from the image, combined with computer vision techniques for extracting features from the segmented objects.

Simulated Attributes

The first method starts from the symbolic scene annotations and transforms these into continuous-valued attributes based on simple rules and procedures. An overview of these rules is provided in Table 4.2. Each symbolic attribute is mapped to one or more continuous attributes with a possible range of values, which can be an interval or a set. For example, colour is mapped to three attributes, one for each channel of the RGB colour space, and size is mapped to a single attribute, namely area. The x-, y- and z-coordinates are taken
4.3. METHODOLOGY

Figure 4.3: Example image from the CLEVR dataset (left) with the corresponding symbolic annotation of a single object (right), namely the large rubber green cylinder. The “3D_coords” and the “pixel_coords” are the coordinates of the object with respect to the 3D rendering environment and the 2D pixel-space, respectively.

Table 4.2: Rules used to transform symbolic object properties to continuous-valued attributes. Note that objects in the CLEVR dataset already have xyz-coordinates.

<table>
<thead>
<tr>
<th>Symbolic</th>
<th>Continuous</th>
<th>Values</th>
<th>Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>colour</td>
<td>R</td>
<td>[0, 255]</td>
<td>± [0, 2]</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>[0, 255]</td>
<td>± [0, 2]</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>[0, 255]</td>
<td>± [0, 2]</td>
</tr>
<tr>
<td>shape</td>
<td>number of sides</td>
<td>{1, 3, 6}</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>number of corners</td>
<td>{0, 2, 8}</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>width-height ratio</td>
<td>[0, 1]</td>
<td>/</td>
</tr>
<tr>
<td>size</td>
<td>area</td>
<td>[0, 100]</td>
<td>± [0, 15]</td>
</tr>
<tr>
<td>material</td>
<td>roughness</td>
<td>[0, 10]</td>
<td>± [0, 2.5]</td>
</tr>
<tr>
<td></td>
<td>x-coordinate</td>
<td>[0, 480]</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>y-coordinate</td>
<td>[0, 320]</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>z-coordinate</td>
<td>[-1, 1]</td>
<td>/</td>
</tr>
</tbody>
</table>

The range of values for the various attributes are not chosen arbitrarily. For colour concepts, I use the RGB value that was used during the image rendering process of the CLEVR dataset\(^1\). This value is used as a seed value and random jitter is added. The same technique is used for the size-related concepts LARGE and SMALL, and for the concepts regarding the objects’ material RUBBER and METAL. The continuous values for material-related concepts are based on a measure of surface roughness. The amount of jitter that is added to each attribute is shown in the rightmost column of Table 4.2. Finally, generating the continuous

\(^1\)This data is available at https://github.com/facebookresearch/clevr-dataset-gen.
attributes for the shape-related attributes proceeds as follows. A sphere is represented as having 1 side, 0 corners and a width/height ratio of 1, a cylinder has 3 sides, 2 corners and a width/height ratio of 0.5 and a cube has 6 sides, 8 corners and a width/height ratio of 1.

Obtaining sensory data in this way is very straightforward and creates a controlled environment. Indeed, even with the presence of random jitter, there is no overlap between different instances of a particular type of concept. Specifically, different colours such as blue and cyan do not overlap in terms of the \( r, g \) and \( b \) features, or large and small do not overlap in terms of the area feature. For each particular type of concept, every instance takes up a disjoint area in the space of continuous-valued attributes. This makes the concept learning task easier and allows to validate the proposed learning mechanisms before moving on to an environment with perceptual processing as could be used by an embodied, autonomous agent.

**Extracted Attributes**

To test the approach using more advanced perceptual processing, a state-of-the-art instance segmentation model is used. It detects and segments the objects directly from the image. After segmentation, a number of numerical attributes is extracted from the proposed regions using computer vision techniques. With this approach, different instances of the same type of concept (such as different colours or shapes) will no longer take up disjoint areas in the attribute space. Additionally, the numerical values will be subject to more noise due to variations in the images such as overlapping objects, lighting conditions or shade effects.

The first step consists of object detection and segmentation. For this, a Mask R-CNN model (He et al., 2017) from the Detectron framework (Girshick et al., 2018) pre-trained on a separately generated set of CLEVR images by Yi et al. (2018) was used. Given an image, the network generates a mask for each object in the scene. A mask is a matrix of the same dimensionality as the image, containing a boolean value for every pixel in the input image, denoting whether or not the pixel belongs to the detected object. All masks with a certainty below 90% are removed. Each mask is multiplied with the original image to obtain a specific highlighted region in that image. For full training regime details of the Mask R-CNN model, I refer to Yi et al. (2018). No separate evaluation of the object detection accuracy is reported.

Next, continuous-valued attributes are extracted from the highlighted regions using computer vision techniques. These attributes are summarized in Table 4.3. As with the previous environment, a number of continuous attributes are provided for each of the symbolic attributes. The mean colour of each region is extracted and represented using the HSV colour space. Afterwards, it is converted to the CIEL*A*B* colour space and split per channel. For shapes, the estimated number of corners is extracted through the Ramer-Douglas-Peucker algorithm (Ramer, 1972; Douglas and Peucker, 1973). This algorithm approximates the
4.3. METHODOLOGY

contour of a region with a similar curve that uses fewer points. The number of points of the curve returned by the algorithm is used as the number of corners. Further, the Hamming distance between the region’s contour and the enclosing circle is used, and the width/height ratio. The size-related attributes are self-explanatory, except for the last two. The bb-area ratio expresses the ratio between the area of the region and the area of the rotated bounding box. The rotated bounding box of a region is a bounding box that is rotated such that the overlap with the region is maximised. Similarly, the image-area ratio expresses the ratio between the region’s area and the area of the entire image. The material of objects is expressed by the ratio of both dark and bright pixels. A pixel is considered dark when it has low brightness (< 20), captured in the ‘V’ dimension of the HSV colour space. Conversely, a pixel is considered bright when it has high brightness (> 70) and low saturation (< 50). The brightness and darkness attributes are based on the idea that the metal objects are more reflective and thus contain more bright pixels. Finally, as before, the x-, y- and z-coordinates are copied over from the symbolic scene annotation.

Table 4.3: Mapping from symbolic attributes to continuous attributes obtained by the image segmentation process.

<table>
<thead>
<tr>
<th>Symbolic</th>
<th>Continuous</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>colour</td>
<td>mean-L</td>
<td>[0, 100]</td>
</tr>
<tr>
<td></td>
<td>mean-A</td>
<td>[−127, 128]</td>
</tr>
<tr>
<td></td>
<td>mean-B</td>
<td>[−127, 128]</td>
</tr>
<tr>
<td>shape</td>
<td>number of corners</td>
<td>R⁺</td>
</tr>
<tr>
<td></td>
<td>hamming distance</td>
<td>[0, 1]</td>
</tr>
<tr>
<td></td>
<td>width-height ratio</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>size</td>
<td>width</td>
<td>R⁺</td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>R⁺</td>
</tr>
<tr>
<td></td>
<td>area</td>
<td>R⁺</td>
</tr>
<tr>
<td></td>
<td>bounding-box area</td>
<td>R⁺</td>
</tr>
<tr>
<td></td>
<td>bb-area ratio</td>
<td>[0, 1]</td>
</tr>
<tr>
<td></td>
<td>image-area ratio</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>material</td>
<td>bright-pixels</td>
<td>[0, 1]</td>
</tr>
<tr>
<td></td>
<td>dark-pixels</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>angle</td>
<td></td>
<td>[0, 180]</td>
</tr>
<tr>
<td>x-coordinate</td>
<td></td>
<td>[0, 480]</td>
</tr>
<tr>
<td>y-coordinate</td>
<td></td>
<td>[0, 320]</td>
</tr>
<tr>
<td>z-coordinate</td>
<td></td>
<td>[−1, 1]</td>
</tr>
</tbody>
</table>
4.3.4 Concept Representation

A concept is represented as a mapping from a symbolic label to a set of continuous-valued attributes. Similar to Wellens (2012), I make use of a weighted set representation where each concept-attribute link has a score \( s \in [0, 1] \), representing the certainty that the given attribute is important for the concept. In contrast to Wellens (2012), the attributes are continuous-valued which enables the use of such concepts in grounded, embodied scenarios. An example concept is shown in Figure 4.4.

![Figure 4.4](image)

Figure 4.4: The concept cube is linked to a weighted set of attributes. The weight represents the certainty of an attribute belonging to the concept. Each attribute is modelled as a normal distribution that keeps track of its prototypical value (i.e. the mean) and the standard deviation. The values between square brackets denote two standard deviations away from the mean. These are not used in similarity calculations directly, but give an indication of the observed range of values.

To computationally operationalize this concept representation in the interaction script outlined in Section 4.3.2, two pieces of functionality are required: (i) a similarity measure between concepts and objects, necessary during conceptualisation and interpretation and (ii) mechanisms for updating concepts on the basis of positive exemplars during alignment.

**Weighted Similarity**

The use of a weighted similarity measure is similar to the adaptive strategies of Wellens (2012), since it allows an agent to use a concept even if it does not exactly match with a particular object. However, Wellens (2012) only considers symbolic attributes, allowing him to implement such a measure using set operations. In this work, a continuous similarity measure is used. Specifically, the similarity between a concept \( C \) and an object \( O \) can be computed by the average similarity between each of their respective attributes, weighted by the certainty of each attribute. Formally, the similarity \( S(C, O) \) is implemented as fol-
4.3. METHODOLOGY

The similarity measure $S$ relies on the similarity measure $S'$ for comparing a concept $C$ and an object $O$ on the level of the attribute $a$. Each attribute within a concept $(C_a)$ is modelled as a normal distribution and the similarity function $S'$ is based on the $z$-score of the attribute value of the object $(O_a)$ with respect to this normal distribution. The $z$-score is additionally embedded in a linear function to transform a small $z$-score in a high similarity value and a large $z$-score in a low similarity value. Specifically, this function maps a $z$-score of 0 to a similarity of 1 and when the $z$-score reaches 2, the similarity has reached 0. By the time the $z$-score would be larger than 4, the similarity is below -1. The similarity measure $S'$ is implemented as follows:

$$S'(C_a, O_a) = \frac{2 - |z(O_a)|}{2}$$

(4.2)

where $z(O_a)$ refers to the $z$-score of the attribute value of the object $O_a$ with respect to the normal distribution for the attribute of the concept $C_a$. Given that the similarity function $S'$ returns a value between -inf and 1 and the scores of attributes are always between 0 and 1, the similarity measure $S$ also returns a value between -inf and 1. In Nevens et al. (2020), the similarity score $S'$ was cut off at -1. This cut-off has been removed in order to be able to rank all objects that are compared to a concept, which will be necessary for integrating the concepts in the visual question answering task later on (see Section 4.6).

The similarity measure $S$ is used in both conceptualisation and interpretation. In conceptualisation, the aim is to find a concept $C$ that is discriminative for the topic $T$. Specifically, the concept $C$ is discriminative when its similarity to the topic $S(C, T)$ is larger than the similarity $S(C, O)$ for any other object $O$ in the scene. If this condition holds for multiple concepts, the concept that maximises the difference in similarity is chosen. Put differently, during conceptualisation, the learner chooses the concept that makes the topic stand out the most among the other objects in the scene. In interpretation, the learner finds the object in the scene that maximises the similarity with respect to the concept it could parse from the tutor’s utterance.

**Concept Alignment**

During alignment, the learner will update the concept that it used during the interaction both in terms of the prototypical values and the certainty scores of the attributes. This
update is based on the feedback provided by the tutor, who points out a positive exemplar. Specifically, the learner will shift the attributes of the concept such that they are closer to this positive exemplar and update the certainty scores of attributes such that the concept can better discriminate the exemplar. This way, the agent can gradually shape its repertoire of concepts to fit the environment and the communicative task. This update procedure works in two steps.

1. **Updating prototypical values.** The agent updates the prototypical values of all the attributes in the concept. The reason for updating all attributes, as oppose to only those with high certainty scores for example, is to allow for flexibility. In particular, when an attribute suddenly becomes important later on in the experiment, e.g. because of changes in the environment, this attribute’s value should also reflect the observed examples. The update mechanism makes use of Welford’s online algorithm (Welford, 1962). This is an online algorithm that specifies recurrence relations for the mean and standard deviation of normal distributions. It allows to recompute the mean and standard deviation on the basis of a single observation, without the need to store all previous observations. Concretely, each attribute keeps track of the number of observations \( N \), the prototypical value \( p_n \) and the sum of squares of differences \( M_{2,n} \) from the current prototypical value. The \( n \) in \( M_{2,n} \) denotes the current interaction. Given a new observation \( x_n \), these values can be updated using the following steps:

\[
\begin{align*}
N &= N + 1 \\
\delta_1 &= x_n - p_{n-1} \\
p_n &= p_{n-1} + \frac{\delta_1}{N} \\
\delta_2 &= x_n - p_n \\
M_{2,n} &= M_{2,n-1} + (\delta_1 * \delta_2)
\end{align*}
\]

The standard deviation, required for computing the \( z \)-score in the similarity measure, can be computed from \( N \) and \( M_{2,n} \) as follows:

\[
\sigma = \sqrt{\frac{M_{2,n}}{N}}
\]

2. **Updating certainty scores.** The agent increases the certainty of the subset of attributes that is most discriminative for the topic. The certainty score is decreased for all other attributes. A subset of attributes is discriminative when it is more similar to the topic than to any other object in the scene. Since this can be true for multiple subsets, the most discriminative subset is defined as the subset where the difference between the similarity to the topic and to the next most similar object is maximised. Thus, during alignment, the agent not only uses the topic object itself, but also compares this to other objects in the
scene. This ensures that the combination of attributes, and ultimately the entire repertoire of concepts, is functionally relevant in the agent’s environment. The similarity functions $S$ and $S'$, defined above, are used to compute the most discriminative subset of attributes. However, to reduce the computational load, not all subsets of attributes are considered. Specifically, all subsets are filtered to contain at least the set of attributes that are discriminative on their own. The procedure to update the certainty scores can be summarised as follows:

- Using the similarity function $S'$, the agent identifies the discriminative attributes. This yields e.g. area and nr-of-corners.

- The agent computes all subsets of attributes of the concept.

- The agent filters all subsets and keeps only those that contain at least the attributes found in the first step. This yields subsets such as \{area, nr-of-corners\}, \{area, nr-of-corners, wh-ratio\}, \{area, nr-of-corners, roughness\}, etc.

- The agent finds discriminative subset(s) of attributes using similarity function $S$.

- The previous step can produce multiple subsets. The agent takes the one that maximises the difference in similarity between the topic and the next most similar object.

- The agent increases the certainty score of the attributes in this subset, and decreases the certainty score of all other attributes.

**Discussion**

While this concept representation is relatively easy to grasp, an important assumption was made in its design, namely that the attribute values are modelled using normal distributions. Statistical testing, using the normality test by D’Agostino and Pearson (d’Agostino, 1971; d’Agostino and Pearson, 1973), has shown that this is not the case for any of the attributes that were generated using the procedures explained in Section 4.3.3. The distributions of the attributes come close to normal distributions but have thinner tails at both ends. Still, this can be viewed as odd, especially for some of the studied concepts. Take the concept **left** as an example. It is important to note that the concept of **left** in this experiment refers to ‘left in the image’ and not ‘left of another object’, although the latter can be derived from the former (see Section 4.6). With this definition of left, the X-coordinate is most likely to be an important attribute for this concept. In the images of the CLEVR dataset, the X-coordinate of an object represents the central point of the object and can be anywhere between 0 and 480. Thus, in theory, an object is considered to be **left** when its X-coordinate is smaller than 240. The bulk of objects that can be considered **left** will not be close to 0, nor close to 240, but somewhere in between, e.g. around X-coordinate 170.

\[2\text{In practice, the objects will be far enough removed from the edges of the images such that they are completely visible.}\]
Using the concept representation as described above, a normal distribution with a mean value around 170 could be learned for the X-coordinates and allow the agent to communicative about objects on the left side of the image. However, one could argue that objects with an X-coordinate smaller than 170 can actually be considered “more left”, while objects with an X-coordinates larger than 170 are gradually “less left”. Currently, this cannot be captured by the concept representation as described above.

Another assumption is made concerning the size-related concepts SMALL and LARGE. In the CLEVR dataset, these concepts are considered in absolute terms. The sizes of the objects are not expressed relatively to each other, but an object is considered to be either small or large. This assumption, together with the closed world of the CLEVR dataset, allows to use the concept representation as described above for these types of concepts. However, in more open domains, a different kind of concept representation would be required, e.g. to capture “the large mouse” or “the small elephant”. In these cases, the size-related concept could learn which attributes of the concept tied to the noun are relevant and how their prototypical values should be shifted. This is currently not captured by the concept representation described above.

4.3.5 Tutor Behaviour

The tutor uses the symbolic scene annotation for both conceptualisation (i.e. acting as the speaker) and interpretation (i.e. acting as the listener). Assume that the topic that can be described symbolically as \{green, cube, large, rubber, left, front\}. In conceptualisation, the tutor will try to describe this object using a single concept. Traversing the concepts of the topic in a random order, the tutor will check if no other objects in the scene share this concept. For example, if the topic is the only cube in the scene, the concept \(\text{cube}\) will be returned. In interpretation, the tutor looks for all object in the scene that possess the attribute that was uttered by the learner. If there is only one object that can be found, interpretation succeeds and the tutor points to this object, indicating that it thinks this object is the topic that was intended by the learner. If multiple objects were found, interpretation stops and the interaction is considered a failure.

4.4 Experimental Setup

This section describes three experiments that are designed to showcase the various desirable properties of the proposed methodology for grounded concept learning. In the first experiment, the baseline performance is established (Section 4.4.1). In subsequent experiments, I test how well the concepts generalise (Section 4.4.2), and how they can be learned incrementally in a changing environment (Section 4.4.3).
4.4 EXPERIMENTAL SETUP

4.4.1 Main Experiment

The first experiment validates the concept learning mechanisms via the language game setup laid out in Section 4.3. The learner’s performance is evaluated using both the simulated features and the noisy features extracted directly from the images (see Section 4.3.3). In both settings, scenes from the validation split of the CLEVR dataset are used. This split consists of 15,000 unique scenes, each containing between three and ten objects. The learner agent is evaluated in terms of communicative success, communicative success given conceptualisation and concept repertoire size (see Section 4.3.2). The goal is to validate whether or not the agent can successfully acquire the concepts from the tutor and use them in bidirectional communication. At the end of the experiment, the acquired concepts can be inspected to see which combinations of attributes were found to be relevant for the object reference task in the CLEVR environment.

4.4.2 Generalisation Experiment

Using the CLEVR CoGenT dataset (Johnson et al., 2017a) (see also Section 3.3.4), it is possible to test if the acquired concepts are general enough to extend to similar, yet unseen, objects and combinations of attributes. The CLEVR CoGenT dataset consists of two conditions. In condition A, cubes can be grey, blue, brown, or yellow, cylinders are red, green, purple, or cyan and spheres can have any of these colours. In condition B, the colour options for cubes and cylinders are swapped and the colour options for spheres remain the same. Like the original CLEVR dataset, the CoGenT data comes with a symbolic annotation that can be transformed into continuous-valued attributes using the two methods described in Section 4.3.3. The goal of this experiment is to validate if the agent truly learns the concepts independently from the statistical distributions or co-occurrences in the environment, which are often exploited in other types of models. This is evaluated by running a number of interactions in condition A, switching off the learning operators, and running a number of interactions in condition B to evaluate the communicative success. When the communicative success remains stable after switching to condition B, this indicates that the concepts acquired by the agent do not rely on attribute co-occurrences from the environment, and that the concepts acquired by the learner are general enough to be used in both conditions even though they were learned in the first condition. Additionally, by varying the number of interactions used for learning in condition A, I gain insight into the speed at which the learner can acquire concepts that are sufficiently functional in the environment.

4.4.3 Incremental Learning Experiment

By incrementally expanding the environment, the adaptivity and open-endedness of the concept learning approach is demonstrated. For this experiment, I created a novel variation of the CLEVR dataset consisting of five splits. In each split, more concepts are added and
less data is available. Specifically, in the first split, there are 10,000 images where all objects are large, rubber cubes in four different colours. In the second split, there are 8,000 images and the cubes can be large or small. Spheres and cylinders are added in the third split and the data is reduced to 4,000 scenes. The fourth split again halves the amount of data and metal objects are added. Finally, in the fifth split, four more colours are added and only 1,000 scenes are available. The splits are summarized in Table 4.4.

Table 4.4: Summary of the concepts in each split of the incremental learning dataset.

<table>
<thead>
<tr>
<th>Split</th>
<th>Concepts</th>
<th>Scenes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GREY, RED, BLUE, GREEN, CUBE, RUBBER, LARGE</td>
<td>10,000</td>
</tr>
<tr>
<td>2</td>
<td>Concepts from split 1 + SMALL</td>
<td>8,000</td>
</tr>
<tr>
<td>3</td>
<td>Concepts from split 2 + SPHERE, CYLINDER</td>
<td>4,000</td>
</tr>
<tr>
<td>4</td>
<td>Concepts from split 3 + METAL</td>
<td>2,000</td>
</tr>
<tr>
<td>5</td>
<td>Concepts from split 4 + BROWN, PURPLE, CYAN, YELLOW</td>
<td>1,000</td>
</tr>
</tbody>
</table>

The agent is exposed to each of the splits consecutively, without resetting its repertoire of concepts or switching off the learning operators. Throughout the experiment, the communicative success and the concept repertoire size are monitored. The goal of this experiment is threefold. First, it shows that the learning mechanisms can easily and quickly adjust to a changing environment and that there is no need to specify the number of concepts that should be learned in advance nor to fully or even partially re-train the repertoire of concepts when new concepts become available, as is often the case for other types of models (see Section 4.2). Second, it demonstrates the transparency of the concept representation. Indeed, as the environment changes, certain attributes might become more or less important. This evolution can be easily monitored throughout the experiment. Third, the experiment shows the data efficiency of the concept learning mechanism by reducing the available number of scenes throughout the splits and still reaching communicative success.

4.5 Experimental Results

This section presents the results of the experiments outlined in Section 4.4. All experiments were ran for ten series of maximum 10,000 interactions and the metrics were averaged, with the error bars showing the 5th and 95th percentile. The plots for the communicative success use a sliding window of 100 interactions. The Babel software package (see Section 2.2.4) was used to implement and run the experiments and plot the results.

4.5.1 Main Experiment

In the first experiment, the learning mechanisms are validated. Specifically, the learner’s ability to successfully acquire the repertoire of concepts through communication is measured in terms of communicative success and concept repertoire size, both in the simulated
4.5. EXPERIMENTAL RESULTS

Figure 4.5: In both environments and across settings, the communicative success rises quickly and converges to a stable level after merely $\sim$1000 interactions. The agent acquires exactly 19 concepts.

Environment and in the more noisy environment. Figure 4.5 provides an overview of the results. A distinction is made between three settings: (i) communicative success when the learner always acts as the listener (solid teal-coloured line), (ii) communicative success where both agents can be speaker or listener (dashed yellow line), and (iii) communicative success given conceptualisation when both agents take on both roles (dotted red line). Figure 4.5a shows these metrics for the simulated environment, while Figure 4.5b reports them for the noisy environment. In both environments and across the three settings, communicative success rises quickly and reaches a stable level after merely $\sim$1000 interactions. Table 4.5 provides an overview of the communicative success that was reached after 5000 of the 10,000 interactions in total. The plots in Figure 4.5 are cut of at this point since all metrics reached a stable level. Just like the communicative success, the concept repertoire size increases rapidly and stabilizes at 19 concepts, which are all concepts present in the agents’ environment.

Table 4.5: Communicative success after 5,000 interactions across the experimental settings and environments (using a sliding window of 100 interactions).

<table>
<thead>
<tr>
<th></th>
<th>Simulated (%)</th>
<th>Noisy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicative success (always listener)</td>
<td>99.7</td>
<td>92.0</td>
</tr>
<tr>
<td>Communicative success (both roles)</td>
<td>98.9</td>
<td>86.5</td>
</tr>
<tr>
<td>Communicative success if conceptualisation (both roles)</td>
<td>99.8</td>
<td>94.4</td>
</tr>
</tbody>
</table>

Error analysis has revealed that the small percentage of failed interactions in the simulated environment is caused by the concepts capturing spatial relation. Without these concepts, communicative success converges to 100% in all three settings after $\sim$500 interactions. As discussed in Section 4.3.4, these spatial relations are in absolute terms, i.e. the concept LEFT denotes ‘left in the image’. Inspection of these concepts’ attributes, such as xpos, ypos
and zpos, shows that the learner acquires prototypes with mean values very close to the center of the image and very wide tails at both ends. This causes the agent to, for example, mistakenly use the concept left whenever the topic is close to the centre of the image, but still on the right side, and the other objects are further to the right in the image.

In the noisy environment, failed interactions cannot be attributed to one particular type of concept. The communicative task is inherently more difficult due to the overlapping feature values, as discussed in Section 4.3.3. Furthermore, the noisy environment clearly illustrates that conceptualisation is a more difficult process than interpretation. When comparing the learner as listener to the learner taking on both roles, the communicative success drops by 5.5 percentage points. However, the communicative success given a successful conceptualisation is 2.4 percentage points higher on average than the success in the listener-only setting. This indicates that the decrease in success between the first and the second setting can be explained by unsuccessful conceptualisations.

The concept representation proposed in this chapter allows for a clear and easy to interpret view on the learned concepts. This is demonstrated in Figure 4.6, showing the concept sphere obtained after 5,000 interactions in both the simulated and noisy environments. In both cases, few attributes are needed to discriminate the concept sphere. The importance of these attributes is reflected by their high certainty scores. In the simulated world, discriminative attributes are number of corners and number of sides, while in the noisy world these are the width/height ratio, the circle-distance and the bb-area-ratio. The circle-distance represents the Hamming distance between the contour of the object and the minimally enclosing circle. The bb-area-ratio represents the ratio between the area of the object and the area of its bounding box. In both environments, all attributes with high scores are indeed intuitively shape-related. An overview of all concepts acquired in a single experimental run in both the simulated world and the noisy world is provided in Appendix A (Figures A.1 and A.2).

![Diagram](image_url)

Figure 4.6: The acquired concepts are human-interpretable and capture discriminative combinations of attributes. The concept sphere focusses on attributes related to shape in both environments. Attributes with certainty score 0 are hidden.

With this experiment, I have shown that the learner agent can distil meaningful concepts
from streams of continuous data in the form of discriminative subsets of attributes and their prototypical values, and is able to successfully use them in communication. Furthermore, as these concepts are expressed using human-interpretable feature channels, both the learning mechanisms and resulting repertoire of concepts are completely transparent.

In presenting the remainder of the experimental results, the distinction between an ‘always listener’ setting and ‘both speaker and listener’ setting will no longer be made. The latter will be used by default, since bidirectionality is one of the key desirable properties of human-like communication systems.

### 4.5.2 Generalisation Experiment

The generalisation experiment demonstrates the agent’s ability to learn the concepts completely independently from the statistical distributions or attribute co-occurrences in the dataset. Using the CLEVR CoGenT dataset, the agent learns concepts during a number of interactions in condition A. Afterwards, the learning operators are turned off and the communicative success of the agents is evaluated in condition B for the remainder of the interactions. When successful, the agents’ level of communicative success should remain stable when transitioning from condition A to B. As before, both the simulated environment and the noisy environment are considered. Additionally, the amount of training interactions in condition A is varied to test the speed at which the learner agent can acquire a functional repertoire of concepts.

![Communicative Success](image)

**Figure 4.7**: Communicative success after learning for the specified number of interactions in condition A. The concepts are learned completely independently from the co-occurrences in the environment. Given at least 1000 interactions in condition A, the agents achieve the same level of communicative success as in the bidirectional setting of the previous experiment.

Figure 4.7 shows the communicative success of the agents during the learning phase (in condition A) and the evaluation phase (in condition B). Figures 4.7a and 4.7b vary in the number of interactions that took place in condition A before transitioning to condition B. From Figure 4.7a, it is clear that the learner agent cannot reach the same level of success as
in the previous experiment after only 500 interactions in condition A. However, with 1000 interactions (Figure 4.7b), this level of success is achieved. This indicates that the learner’s repertoire of concepts is shaped rapidly and sufficiently to have successful interactions. When transitioning from condition A to B, there is no decrease in communicative success in the simulated environment and only a minor decrease in the noisy environment. This indicates that the concepts acquired by the agent abstract away over the observed instances.

To further investigate the generalisation abilities of the learner, the acquired concepts are examined. Remember that in condition A in the CoGenT dataset, cubes can be grey, blue, brown, or yellow, cylinders have a set of different colours and spheres can be any colour. Figure 4.8 shows the concept representation of the colours for cubes after being learned in condition A for 1000 interactions in the simulated environment. If the agent would rely on co-occurrences of the dataset, the concept representation of these colours could contain attributes related to shape, since each time one of these colours occurs it is either a cube or a sphere. Additionally, cubes and spheres have the same value for the wh-ratio attribute.
so it could be considered discriminative. As can be seen in Figure 4.8, even though this feature is present in the concept yellow, its certainty score is very low. The other colour concepts only focus on feature channels $r$, $g$ and $b$. Hence, the agent does not focus on particular dataset co-occurrences and is able to generalize over various observations. The same is true for the shape-related concepts cube, cylinder and sphere. These are not distracted by the fixed colour options of condition A and the switch in colour options in condition B. These concepts are shown in Appendix A (Figures A.3 and A.4). This can be attributed to the notion of discrimination, which makes sure that only relevant attributes obtain a high certainty score.

### 4.5.3 Incremental Learning Experiment

The proposed methodology for grounded concept learning is completely open-ended and has no problems dealing with a changing environment. This is validated through the incremental learning experiment where, over the course of 15,000 interactions, the number of concepts in the environment increases. The amount of interactions before new concepts are introduced is varied between 500 and 1000 interactions. The learning mechanisms are able to adjust almost instantly to these changes, as is shown in Figure 4.9. In both environments, the communicative success drops when transitioning from one phase to the next. However, the agent quickly acquires new concepts and ultimately reaches the same level of communicative success as the main experiment of Section 4.5.1.

Figure 4.9: Communicative success in the incremental learning experiment. New concepts are introduced every 500 interactions (4.9a) or 1000 interactions (4.9b). The learning mechanism is completely open-ended, allowing the agent to adapt to a changing environment. Note that the x-axis of 4.9b is different from 4.9a to best show the changes in communicative success.

The concepts in the incremental learning experiment have relevant attributes with high certainty scores already after the first phase of the experiment (see Figure 4.10). Consequently, these remain stable throughout the various other phases, while other attributes come and go but never achieve high certainty scores.
The concepts at the end of the experiment have the same high-scoring attributes as those obtained in the first experiment, independent of the phase in which they were introduced. This is illustrated in Figure 4.11.

![Diagram](image)

Figure 4.10: The concept grey after each of the five phases (1000 interactions per phase) in the noisy environment. The relevant attributes obtain a high certainty score after the first phase of the experiment.

![Diagram](image)

Figure 4.11: The final representation of three concepts introduced during various phases of the experiment (1000 interactions per phase) in the noisy environment. The concept blue was introduced in phase 1, cylinder in phase 3 and rubber in phase 4.

### 4.6 Fully Explainable Visual Question Answering

In this section, the symbolic repertoire of grounded concepts is evaluated through a higher-level reasoning task. Specifically, I integrate these concepts as semantic entities in the procedural semantic representations of the CLEVR grammar (see Section 3.4.2) and tackle the CLEVR benchmark task. This constitutes a system that is end-to-end transparent, explainable and human-interpretable, ranging from the perception and categorisation of objects
4.6. FULLY EXPLAINABLE VISUAL QUESTION ANSWERING

via the methodology presented in this chapter to the constructional language processing of the question and symbolic reasoning via procedural semantics, both part of the CLEVR grammar (Section 3.4). In Section 4.6.1, I describe this integration and the changes that needed to be made. Afterwards, in Section 4.6.2, the results on the CLEVR benchmark task are presented and discussed.

4.6.1 Integration with CLEVR Grammar

The integration of the acquired concepts in the CLEVR grammar requires three modifications. First, the symbolic conceptual inventory of the agent, containing all concepts and categories present in the CLEVR dataset, is replaced by the repertoire of concepts acquired via the experiment outlined in Section 4.5.1. The agent is provided with information on which concepts belong to the same conceptual category. In other words, the agent knows that the concepts of blue, red, green, etc. belong to the conceptual category colour. Learning this aspect during the language game experiment by, for example, grouping together similar concepts on the basis of their attributes could be part of future work. The acquired grounded concepts are now the semantic entities used in the meaning networks, introduced via bind statements. Second, the primitive cognitive operators no longer operate over the symbolic scene annotation. Instead, the objects in the scenes are transformed to sets of continuous-valued attributes using the methods outlined in Section 4.3.3. In terms of IRL, this constitutes the semantic entities that make up the world model (see Section 2.4.3). Third, the primitive cognitive operators needed to be altered in order to operate over the newly instantiated conceptual inventory and world model. In the paragraphs that follow, the alterations that were made to each primitive operation are discussed in more detail. Primitive operations that are not listed here could be used without changes, or only minor modifications such as changing the type information of the arguments. The complete list of primitive operations can be found in Section 3.4.2.

**Filter Primitive**

One mode of operation for the (filter ?output-set ?input-set ?concept) primitive is to filter an ?input-set on the basis of a given ?concept and bind the resulting set to ?output-set. Now, the variable ?concept is bound to a concept using the weighted set representation as introduced in Section 4.3.4 and the ?input-set contains a set of objects as described in Section 4.3.3. To operationalise a filtering operation, the weighted similarity measure \( S \) described in Section 4.3.4 is used. Specifically, for every object in the ?input-set, the weighted similarity between this object and every concept of the conceptual category of ?concept is computed. Each object gets assigned the concept with the highest similarity. The set of objects that has ?concept as the most similar concept are bound to ?output-set. As an example, consider a filtering operation using the concept sphere, applied to a set of three objects denoted \( \{o1, o2, o3\} \) for simplicity. For every object in the ?input-set, the weighted similarity to sphere, cube and cylinder is com-
puted. When objects ‘o1’ and ‘o3’ are most similar to sphere, this set of objects is bound to ?output-set.

**QUERY Primitive**

In the (query ?concept ?object ?category) primitive, in order to query a particular ?category, such as colour, from a given ?object, the weighted similarity \( S \) between the ?object and all concepts of ?category is computed. The concept with the highest similarity is bound to ?concept.

**SAME Primitive**

The (same ?set ?object ?category) primitive is a combination of the query primitive and the filter primitive described above. First, the query primitive is used to obtain a concept, of type ?category, of the ?object. Next, the filter primitive is used to obtain the set of objects with the same concept, excluding ?object.

**RELATE Primitive**

The (relate ?set ?object ?spatial-relation) primitive applies a spatial transformation with respect to ?object in order to apply the ?spatial-relation that was learned in absolute terms, e.g. left in the image, in relative terms, e.g. left of ?object. Specifically, the spatial relation learned in absolute terms has the centre of the image as its reference point, both in terms of the x-, y- and z-coordinates. First, this reference point is shifted such that it becomes the x-, y- and z-coordinate of ?object. This shift is applied to all objects in the scene and to all spatial relation concept. Afterwards, similar to the filter and query primitives, the weighted similarity measure \( S \) is computed for each pair of (shifted) objects and (shifted) spatial relations. The set of objects that has the ?spatial-relation as the most similar concept is bound to ?set. Because of the spatial transformation that was applied, this ?set contains the objects that have the given ?spatial-relation with respect to ?object.

### 4.6.2 Results

The CLEVR grammar, with the acquired repertoire of concepts integrated as semantic entities, is evaluated on the CLEVR benchmark task. Specifically, the evaluation was ran over the validation split of the dataset, consisting of 150,000 questions over 15,000 scenes. Results are averaged over ten independent runs where each run uses a different repertoire of concepts learned over 10,000 interactions. In Section 3.4.4, I concluded that the symbolic variant of the CLEVR grammar achieves 100% accuracy. Thus, any errors in the current evaluation are caused by the integration of the grounded concept representation.
Table 4.6 presents the results in terms of precision, recall and F1-score, weighted according to the answer distribution. The weighted precision corresponds to the percentage of correctly answered questions whenever the VQA system could effectively produce an answer. Indeed, evaluation of a semantic network may fail when the execution of a primitive operators does not return any new bindings. This highlights the transparency and interpretability of the reasoning process, as one can easily retrace the execution of the various primitive operators and inspect why no answer was produced. The weighted recall corresponds to the question answering accuracy, allowing to compare these results to other state-of-the-art models where only accuracy is reported. The models listed in this table employ either modular neural network approaches or neuro-symbolic approaches. These approaches will be discussed in greater detail in Section 5.2, as they are more closely related to the topic of that chapter. The main message here is that these approaches suffer from many of the shortcomings discussed in Section 4.2. In particular, they rely on black-box architectures that consume huge amounts of training data, they capture concept representations in some latent non-interpretable space and they can be deceived by statistical biases from the dataset. In comparison, the approach presented in this chapter is fully transparent, can be learned in a data-efficient manner, generalises well to similar, yet unseen, instances and allows for an open-ended repertoire of concepts, as demonstrated in the experimental results in Section 4.5.

Table 4.6: Evaluation results on the CLEVR benchmark task.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision (weighted)</th>
<th>Recall (weighted)</th>
<th>F1-score (weighted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMN(^3) (Andreas et al., 2016a)</td>
<td></td>
<td>72.1</td>
<td></td>
</tr>
<tr>
<td>IEP (Johnson et al., 2017b)</td>
<td></td>
<td>96.9</td>
<td></td>
</tr>
<tr>
<td>N2NMN (Hu et al., 2017a)</td>
<td></td>
<td>83.7</td>
<td></td>
</tr>
<tr>
<td>Stack-NMN (Hu et al., 2018)</td>
<td></td>
<td>96.5</td>
<td></td>
</tr>
<tr>
<td>MAC (Hudson and Manning, 2018)</td>
<td></td>
<td>98.9</td>
<td></td>
</tr>
<tr>
<td>TbD (Mascharka et al., 2018)</td>
<td></td>
<td>99.1</td>
<td></td>
</tr>
<tr>
<td>NS-VQA (Yi et al., 2018)</td>
<td></td>
<td></td>
<td><strong>99.8</strong></td>
</tr>
<tr>
<td>NSCL (Mao et al., 2019)</td>
<td></td>
<td></td>
<td>98.9</td>
</tr>
<tr>
<td>CLEVR grammar + concepts (simulated)</td>
<td>99.4</td>
<td>96.2</td>
<td>97.8</td>
</tr>
<tr>
<td>CLEVR grammar + concepts (noisy)</td>
<td>70.2</td>
<td>29.1</td>
<td>40.0</td>
</tr>
</tbody>
</table>

In the simulated environment, near perfect precision (99.4%) and a level of accuracy (i.e. weighted recall) that is competitive with several state-of-the-art approaches is achieved. A detailed error analysis has revealed that the lowest accuracy is achieved in questions that have the `INTERSECT` (89.4%), `EQUAL` (94.2%) and `RELATE` (92.1%) primitives in their underlying meaning representation. The `INTERSECT` and `EQUAL` primitives mostly occur in longer questions, requiring many steps to answer. This indicates that errors are caused by the

\(^3\) Evaluation of the NMN model on the CLEVR dataset was carried out and reported by Mao et al. (2019).
CHAPTER 4. LEARNING CONCEPTS THROUGH DISCRIMINATION

propagation of inaccuracies in the concept representations through the meaning network. Of course, with longer meaning networks, more possible points of failure are introduced. The errors caused by the \textsc{relate} primitive corresponds with the findings of the baseline experiment in Section 4.5.1, in that the attribute representation using normal distributions have some difficulty in modelling concepts that capture spatial relations.

In the noisy environment, the VQA system struggles to achieve high levels of accuracy (i.e. weighted recall). This can be attributed to various factors, including the propagation and accumulation of errors in the acquired concepts, or difficulties in capturing concepts, especially spatial relations, using normal distributions. These results also indicate that feature extraction plays a crucial role in the acquisition of concepts and their integration in reasoning systems. Indeed, given that the methodology proposed in this chapter has been shown to achieve high levels of communicative success and question answering accuracy in the simulated environment, a different set of features or more accuracy feature extraction might achieve the same or higher levels. Further analysis in this environment has revealed that in 84% of the questions that are considered incorrect, the system actually did not produce an answer. This is reflected through the weighted precision of 70.2%, which indicates that whenever the system can produce an answer, it is fairly accurate at doing so. As opposed to neural network-based approaches, this VQA system thus abstains from guessing answers based on observed distributions, but only returns an answer when it can infer one. However, a guessing mechanism based on answer distribution could be easily added to boost the system’s performance.

4.7 Conclusion

In order to communicate and reason about their environment, autonomous agents must be able to abstract away from low-level, sensorimotor data streams. They therefore require an abstraction layer that links sensorimotor experiences to high-level, symbolic concepts that are meaningful in the environment and for the task at hand. A repertoire of such concepts provides the necessary building blocks for achieving success in higher-level cognitive tasks, such as communication, reasoning or action planning. Therefore, these concepts should be applicable both when acting as a speaker and as a listener. Similar to how humans can grasp a concept after only a few exemplars, an autonomous agent should acquire these concepts quickly and with relatively little data. Acquired concepts should be general enough to extend to similar, yet unseen, situations and the learning methodology should be adaptive and allow for incremental learning in order to support a changing environment or the introduction of new concepts. Finally, to truly understand the reasoning processes of an autonomous agent, its learning mechanisms and representations should be fully transparent and interpretable in human-understandable terms.

The task of grounded concept learning has been considered in various sub-domains of AI,
as discussed in Section 4.2. Deep learning approaches, for example, offer a very powerful paradigm to extract concepts from raw perceptual data, achieving impressive results but thereby sacrificing data efficiency and model transparency. Version space learning offers a more interpretable model but has difficulties in handling noisy observations. Most similar to the approach presented in this chapter is work from the robotics literature, considering tasks such as perceptual anchoring and affordance learning. However, these tasks mostly focus on a single robot that passively extracts concepts from observations of its environment. In contrast, I argue for an interactive learning approach through the language game paradigm, as presented in Section 4.3. The notion of discrimination plays a central role in forming the repertoire of concepts, thereby ensuring the generality and adaptivity of the concepts such that they are relevant in the agent’s environment. Additionally, the proposed methodology offers an explainable concept representation, acquired through a data efficient and incremental method. In Section 4.4, dedicated experiments were set up to highlight each of these desirable properties. In sum, this chapter has presented a novel, discrimination-based approach to learning meaningful concepts from streams of sensory data. For each concept, the agent finds discriminative attribute combinations and their prototypical values. The experimental results in Section 4.5 have shown that these concepts (i) can be acquired quickly with relatively few data points, (ii) generalise well to unseen instances, (iii) offer a transparent and human-interpretable insight in the agent’s memory and processing, and (iv) are adaptive to changes in the environment. These properties are highly valuable in the domains of robotics and interactive task learning, where interpretability, open-endedness and adaptivity are important factors. Once a repertoire of symbolic concepts, abstracting away over the sensorimotor level, has been acquired, the autonomous agent can use it to tackle higher-level reasoning tasks such as navigation, (visual) dialogue and action planning. This was demonstrated for the task of visual question answering on the CLEVR benchmark task in Section 4.6. The evaluation results on this task have brought to light the importance of the feature extraction procedure, but have also demonstrated the capabilities of the approach by achieving results that are competitive with the state-of-the-art, while additionally highlighting the transparency of the reasoning component. Namely, the system does not guess answers when it is uncertain, but instead allows to inspect the various reasoning steps in order to understand why no answer could be formulated.

In order to ensure that the acquired concepts are human-interpretable, the methodology starts from a predefined set of human-interpretable features that are extracted from the raw images. While I argue that this is necessary to achieve true interpretability, it can also be seen as a limitation inherent to the methodology. However, this limitation cannot be lifted without losing the interpretability that the method brings. In the following chapter, a different methodology for concept learning that does not rely on such predefined features is explored. However, as a consequence, it will become more difficult to explain the agents’ internal representations and interpret their reasoning processes.
4.7.1 Contributions

This chapter has presented two contributions of this dissertation. The first contribution consists of a novel concept representation and learning mechanism that allows an autonomous agent to extract meaningful concepts from its sensorimotor experiences through task-oriented communicative interactions (C2). In line with the main objective of this dissertation (O1), the presented methodology focuses on key properties found in human communication systems, such as adaptivity, robustness, bidirectional processing and learning efficiency. This methodology not only contributes to the research on the emergence of concepts within the language game paradigm, it is also directly applicable and highly relevant for research in robotics and interactive task learning due to the aforementioned properties. The latter is additionally demonstrated by the second contribution of this chapter, namely the integration of grounded concepts in a higher-level reasoning task (C3). Applied specifically to the task of visual question answering, all components in the presented system are transparent and human-interpretable and ‘inherit’ the desirable properties of the grounded concept learning methodology. This directly contributes to the secondary objective of this dissertation (O2).
Chapter 5

Learning Concepts as Neural Modules

5.1 Introduction .................................................. 146
5.2 Related Work ................................................... 147
  5.2.1 Modular Neural Networks .............................. 148
  5.2.2 Neuro-Symbolic Approaches ......................... 153
  5.2.3 Discussion .................................................. 154
5.3 Methodology .................................................... 155
5.4 Experimental Setup .......................................... 156
  5.4.1 Primitive Operators ..................................... 156
  5.4.2 Module Architectures .................................... 164
  5.4.3 Training ..................................................... 168
  5.4.4 Neural Modules Interface .............................. 170
5.5 Experimental Results ....................................... 173
  5.5.1 Neural Module Evaluation ......................... 173
  5.5.2 CLEVR Benchmark ..................................... 175
  5.5.3 Generalisation Experiment ......................... 177
5.6 Conclusion ....................................................... 179
  5.6.1 Contributions .............................................. 181
5.1 Introduction

In the previous chapter, I presented a discrimination-based methodology for grounded concept learning where an agent learns to extract symbolic concepts from continuous streams of sensorimotor data. Specifically, through situated, task-oriented, communicative interactions, the agent simultaneously extracts discriminative attribute combinations and their prototypical values from continuous data streams in order to construct a repertoire of concepts that is functional in its environment and for the given task. This methodology posits a number of properties which are highly desirable in the domains of robotics and interactive task learning, specifically its interpretability, open-endedness and adaptivity. Next to these properties, the experimental results have shown that high levels of communicative success could be reached with relatively few observations. However, it was also clear from these results that a noisy environment with more realistic perceptual features proved to be more challenging, leaving room for improvements. This became especially clear when the concepts acquired in this environment were integrated in the CLEVR benchmark task. Concretely, the question answering accuracy degraded when an increasing number of concepts is required to answer the question and their respective inaccuracies accumulate. Finally, as noted at the end of the previous chapter, the interpretability of the approach relies on a set of predefined human-interpretable features that are extracted from the raw images on beforehand.

In this chapter, I introduce a different approach to grounded concept learning. Specifically, the concept learning task is now embedded within the VQA task through hybrid procedural semantics. Hybrid procedural semantics follows the main idea of procedural semantics, namely that the meanings underlying linguistic utterances are captured in programs that can be executed algorithmically (see Section 2.4.1), but extends this such that the programs feature a combination of symbolic and sub-symbolic operations. The sub-symbolic operations are implemented through one or more modular neural networks, also called (neural) modules, which operate over continuous data streams, namely images, and produce either image masks, i.e. highlighted regions within the image, or symbolic labels. Crucially, each such module captures a particular concept. For instance, a module that captures the concept blue focusses exclusively on recognizing things in images that are blue, regardless of the objects’ other features. Consequently, training these modules on raw image data corresponds to learning the corresponding concepts and grounding them in the images. Through the integration of neural modules in procedural semantic representations, hybrid procedural semantics combines the strengths of sub-symbolic techniques, namely pattern recognition on unstructured data, with those of symbolic techniques, namely higher-level reasoning on structured data. It allows for information to be shared between the sub-symbolic level and the symbolic level in an elegant and highly flexible manner, steered by the linguistic analysis of the utterance that led to the procedural semantic representation. This novel methodology relies on ideas from procedural semantics (Woods, 1968; Wino-
5.2. RELATED WORK

In this section, I introduce related work on the task of visual question answering (VQA). Particularly, I focus on two approaches that have been predominantly used for this task over the last five years. These are the modular neural networks approaches (Section 5.2.1) and the neuro-symbolic approaches (Section 5.2.2). So-called monolithic approaches, i.e. models that consist of a single, large neural network trained in an end-to-end manner, were the first to tackle VQA tasks (see e.g. Gao et al. (2015); Ren et al. (2015) and Ma et al. (2016)). However, these models are not considered in this section. Ever since the inception of modular neural networks and neuro-symbolic models, monolithic models are outperformed not only in terms of accuracy but particularly in terms of interpretability and generalisation abilities. These and other properties are exactly what I set out to achieve...
in truly intelligent systems (see Section 1.2), making monolithic models irrelevant for the purposes of this dissertation.

5.2.1 Modular Neural Networks

All models following the modular neural networks approach operate according to the same basic outline. One component, called the program generator, maps the natural language question onto a layout of neural modules. Another component, called the execution engine, connects the neural modules as dictated by the program generator such that a single, large neural network is formed. The execution engine is typically fed with a high-dimensional encoding of the input image. These encodings are obtained by feature extractors, such as LeNet (LeCun et al., 1989), VGGNet (Simonyan and Zisserman, 2014) or ResNet (He et al., 2016), pretrained on huge datasets of annotated images, such as ImageNet (Deng et al., 2009) or MS-COCO (Lin et al., 2014). Consequently, the representations that are passed in between modules typically consist of high-dimensional convolutions or attentions over these image encodings, making it difficult to interpret intermediate results and the overall reasoning (i.e. question answering) process. In most cases, different types of modules with specific neural architectures are designed depending on the task they need to perform. For instance, distinct neural architectures are required for finding all objects of a particular colour or for finding all objects that have a particular spatial relation with respect to another. Modules are typically trained through the execution engine. In other words, the network of modules assembled by the execution engine is end-to-end differentiable and updated with backpropagation as if it were a single neural network. Finally, modules typically capture a range of tasks (or concepts) and their behaviour can be conditioned on the basis of predefined embeddings, such as GloVe (Pennington et al., 2014), or learned embeddings that capture part of the input question. For example, a single module can be used to find objects of various colours, conditioned using the word embedding of one specific colour.

State of the Art

Andreas et al. (2016b) were the first to combine semantic parsing with neural networks for visual question answering. Their Neural Module Networks (NMN) model uses an off-the-shelf parser (Klein and Manning, 2003) in combination with a set of hand-written rules to determine the composition of modules from the input question. This allows the modules to be trained on the basis of (question, image, answer) triples alone. A small set of modules was designed by the authors, each with a specific functionality and corresponding architecture.

By the same authors, the D-NMN model (Andreas et al., 2016a) improves over NMN in two ways. First, the structure of the module network is learned by passing both an LSTM encoding of the question and a feature vector representing the dependency parse of the
question through a multilayer perceptron. This replaces the hand-crafted rules of the NMN model. Second, the D-NMN model is applied to both images and structured knowledge bases, using the same set of reusable modules. Similar to NMN, D-NMN only requires (question, image, answer) triples.

Trading data efficiency for improved accuracy, Johnson et al. (2017b) propose two major changes compared to D-NMN. First, they get rid of the external parser that is required to find the layout of the module network. Instead, the network structure is predicted by passing the question through an LSTM that returns a sequence expressing the prefix traversal of the network. While both the program generator and the execution engine are now learned end-to-end, this comes at a cost. Unlike previous models, an additional dataset of questions annotated with ground-truth network layouts is required for training the LSTM. A second novelty is automatic module specialisation. Johnson et al. (2017b) did not design specific architectures for specific types of modules. Instead, all modules use the same generic architecture and learn to specialise on their respective task through the joint training procedure. As a consequence of this architecture, convolutions are passed in between modules, as opposed to attentions. In what follows, I will refer to this model as IEP, which is a shorthand for the title of the paper by Johnson et al. (2017b).

The model by Hu et al. (2017a), called N2NMN, extends both IEP and D-NMN. Similar to IEP, the entire model can be learned end-to-end. Similar to D-NMN, different attention-based module architectures are used. The main novelty proposed by Hu et al. (2017a) is the incorporation of a soft attention over question words provided to each module. This allows the model to learn the word embeddings that are used to condition the modules’ behaviour, instead of using fixed, predefined embeddings, as is done in IEP.

Building on both IEP and N2NMN, Hu et al. (2018) no longer require a dataset of questions annotated with ground-truth network layouts. Instead, their Stack-NMN model predicts a distribution over modules and produces attentions over the input question to steer the modules’ behaviour. Consequently, their model no longer makes a discrete decision on the module layout, but a continuous one, allowing also this part of the model to be learned. A differentiable stack data structure is added to store and retrieve intermediate module outputs. Different module architectures are used, as in N2NMN.

Hudson and Manning (2018) propose a novel, generic architecture for solving VQA tasks: the Memory, Attention and Composition (MAC) network. A MAC network is made up of a number of MAC cells representing general-purpose, attention-based reasoning components. A MAC cell explicitly separates control from memory, where the latter consists of a read unit and a write unit. The control unit attends to part of the input question, thereby specifying the operation that the MAC cell needs to perform. Based on that, the read unit extracts relevant information from the encoded input image. The write unit aggregates the result of the operation into the memory structure, such that it can be read by the next MAC cell. Encodings of both the input question and the input image are used throughout
the MAC network. Through their general-purpose design, MAC cells infer their behaviour from the input data. Consequently, no explicit layout of MAC cells needs to be generated as training data. Instead, the cells are always structured in a sequence with the length of the sequence being a hyper-parameter that can be optimised. Additionally, MAC networks are very data efficient, requiring 5x less data compared to the previously discussed models to achieve competitive results.

In the same year, Mascharka et al. (2018) extended both IEP and N2NMN in a different direction. Specifically, they combine the LSTM-based layout prediction of IEP with the attention-based modules from N2NMN. The architectures of the modules are altered such that the intermediate outputs have the same dimensionality as the original image. In other words, they produce image masks instead of high-dimensional attentions over encoded image features. This allows to directly visualise and inspect the reasoning steps that are taken during the execution of the module network, offering a level of transparency and interpretability that goes well beyond previously discussed models. At the same time, they achieve state of the art performance of 99.1% on the CLEVR dataset. This model is named Transparency By Design (TbD).

Focussing specifically on systematic generalisation, Bahdanau et al. (2019) developed another modification of the IEP model. Systematicity is the ability of humans to recombine known skills in previously unseen combinations (Fodor and Pylyshyn, 1988). In the context of VQA, this corresponds to the evaluation of a composition of modules at test time which was not seen during training. As shown by Bahdanau et al. (2019), various modular neural networks approaches struggle with this task. In response, Bahdanau et al. (2019) altered the generic module architecture of IEP such that the intermediate representations are now vector-valued, as oppose to tensor-valued. Through this modification, they obtain state-of-the-art results on the CoGenT generalisation test and on their own CLOSURE dataset. The latter specifies a number of questions that were specifically designed to test systematic generalisation by extending the possible combinations of modules beyond what is available in the CLEVR and CoGenT datasets. As in IEP, annotated questions were required for training the program generator and the vector-valued intermediate representations are not very transparent. This model is referred to as Vector-NMN.

Recently, the Vector-NMN model was outperformed in terms of systematic generalisation by Yamada et al. (2022). They introduced Transformer Module Networks (TMN), using the Transformer architecture as the basis for modules. Other parts of the model, such as the program generator, are taken over from IEP and Vector-NMN. TMN achieves better accuracy both in-distribution and in systematic generalisation scenarios on the CoGenT dataset as well as on the CLOSURE dataset.

Table 5.1 provides an overview of the aforementioned models, comparing them on the criteria that were used throughout this section. In Section 5.3, the same criteria will be discussed for the hybrid procedural semantics approach proposed in this chapter.
Table 5.1: Comparison of models following the neural module networks approach for Visual Question Answering.

<table>
<thead>
<tr>
<th>Question to Module Layout</th>
<th>Module Architecture</th>
<th>Network Input</th>
<th>Intermediate Representation</th>
<th>Module Conditioning</th>
<th>Annotated Questions Required</th>
<th>Training Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMN (Andreas et al., 2016b)</td>
<td>Parser and Rules Parser, LSTM and MLP</td>
<td>Task-specific VGGNet features</td>
<td>Attentions Predefined Embeddings</td>
<td>No End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D-NMN (Andreas et al., 2016a)</td>
<td>LSTM and MLP</td>
<td>Task-specific VGGNet features</td>
<td>Attentions Predefined Embeddings</td>
<td>No End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEQ (Johnson et al., 2017b)</td>
<td>LSTM</td>
<td>Generic ResNet features</td>
<td>Convolutions Predefined Embeddings</td>
<td>Yes End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N2NMN (Hu et al., 2017a)</td>
<td>LSTM</td>
<td>Task-specific VGGNet features</td>
<td>Attentions Predefined Embeddings</td>
<td>Yes End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack-NMN (Hu et al., 2018)</td>
<td>BiLSTM and Controller</td>
<td>Task-specific ResNet features</td>
<td>Attentions Learned Embeddings</td>
<td>No End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC Networks (Hudson and Manning, 2018)</td>
<td>/ MAC cell</td>
<td>MAC cell BiLSTM and ResNet features</td>
<td>Attentions / No End-to-End</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TbD (Mascharka et al., 2018)</td>
<td>LSTM</td>
<td>Task-specific ResNet features</td>
<td>Image Masks Predefined Embeddings</td>
<td>Yes End-to-End</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vector-NMN (Bahdanau et al., 2019)</td>
<td>LSTM</td>
<td>Generic ResNet features Vectors</td>
<td>Predefined Embeddings Yes End-to-End</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMN (Yamada et al., 2022)</td>
<td>LSTM</td>
<td>Transformer ResNet features</td>
<td>Attentions Predefined Embeddings Yes End-to-End</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid Procedural Semantics</td>
<td>CCxG</td>
<td>Task-Specific Raw Image Image Masks / No Independent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 RELATED WORK
Model Properties

At the time of writing, the model by Mascharanka et al. (2018) is the modular neural network approach obtaining state-of-the-art results on the CLEVR dataset (Johnson et al., 2017a). In the following paragraphs, I discuss two approaches applied to the CLEVR dataset that have not achieved state-of-the-art results, but that are relevant in light of the hybrid procedural semantics methodology.

First, Castillo-Bolado et al. (2021) devised a novel training methodology where neural modules can be trained independently from each other, without the need to generate module-specific training datasets. They introduce the concept of a surrogate gradient module, which can be used to train modules for which no direct supervision is available. This surrogate module bridges the gap from the output of the module that needs to be trained to the ground-truth label. Their results indicate that independent training improves compositional behaviour and systematic generalisation.

Second, D’Amario et al. (2021) investigate the specialisation of neural modules in VQA. Specifically, they compare three levels of specialisation. The first level consists of modules which can be conditioned on any type of input, e.g. colours and shapes and spatial relations and so on. In the second level, modules can be conditioned on input categories, e.g. one module for colours and one for shapes and so on. In the third level, modules are even further specialised to the point that no conditioning is available. Instead, separate modules are available for every concept, as in one module for blue, one for cube, one for left and so on. Apart from specialisation, D’Amario et al. (2021) distinguish three types of modules. These are image encoders, intermediate modules and classifier modules for producing the final answer. In all approaches discussed thus far, the image encoders operate on the first level of specialisation, while both intermediate modules and classifiers operate on the second level of specialisation. D’Amario et al. (2021) find that improvements in systematic generalisation can be obtained by also using image encoders at the second level of specialisation.

Neural Modules in Other Tasks

Although the modular neural networks approach originates from work on natural language interfaces for structured data querying and was first applied to VQA, its use has not remained limited to this task. For instance, Manhaeve et al. (2018, 2021) have integrated neural modules in probabilistic logic programs in a way that, very much like hybrid procedural semantics, combines the strengths of both paradigms. They demonstrate their approach through logic predicates that can operate directly on the digit images from the MNIST dataset (LeCun et al., 1998). Another domain where neural modules have been applied is in visual grounding or visual reference, where the goal is to ground (or localise) utterances and parts of utterances in images, e.g. as in Hu et al. (2017b), Liu et al. (2019a), and Subramanian et al. (2020). In robotics, modular neural networks were used to facili-
5.2. RELATED WORK

State transfer learning between tasks and between robots (Devin et al., 2017). Finally, the task of visual dialogue, introduced by Das et al. (2017a), is a direct extension of VQA. Consequently, approaches that have worked well for VQA have been extended for visual dialogue, such as MAC networks (Hudson and Manning, 2018; Shah et al., 2020) and NMN (Kottur et al., 2018; Cho and Kim, 2021).

5.2.2 Neuro-Symbolic Approaches

Neuro-symbolic approaches for visual question answering typically consist of three components. Two neural network models are used to analyse the image and the utterance, respectively. The third component is a symbolic execution engine that is steered by the result of the utterance analysis and operates over the result of the image analysis. The benefit of these approaches is that they combine the strengths of symbolic and sub-symbolic techniques. Specifically, neural network techniques are applied to find patterns in unstructured input, namely images and utterances, while reasoning over these patterns is performed by symbolic techniques. An additional benefit of this approach, in particular of the symbolic execution, is its level of transparency. The reasoning process becomes easier to interpret and reasoning errors can be diagnosed more rapidly.

Yi et al. (2018) were the first to propose a neuro-symbolic approach for VQA. Their NS-VQA model first ‘de-reners’ the image into a structured scene representation. This representation can best be described as a table containing the properties (i.e. colour, size, shape and material) and 3D coordinates of every object. A Mask R-CNN model (He et al., 2017) was used to detect and segment the objects from the image and to predict their properties simultaneously, while a ResNet model (He et al., 2016) was used to extract the 3D coordinates. To train both of these models, the authors generated a dataset of 4,000 new CLEVR images where each object is highlighted on the pixel level and symbolically annotated with its properties. The second part of the NS-VQA model consists of a BiLSTM for mapping the question onto its underlying logical structure. As in IEP, this structure is expressed as the prefix traversal of the abstract syntax tree of the program that needs to be executed. Execution is taken care of by purely symbolic functions. The logical structure determines which functions need to be executed and the structured scene representation is provided as input. Given that both neural elements can be trained in advance and the execution is handled symbolically, NS-VQA requires much less training data compared to the end-to-end training paradigm used in modular neural network approaches. Indeed, only 4,000 images and 270 questions annotated with programs were used. With few data, state of the art accuracy of 99.8% was achieved on the CLEVR dataset.

The NS-VQA model was extended by Mao et al. (2019) such that it is fully differentiable and thus can be learned end-to-end. Instead of producing a symbolic table of object properties, their perception component detects the objects in the image and represents each of them as an embedding. As in NS-VQA, a Mask R-CNN (He et al., 2017) and a ResNet model (He
et al., 2016) are used. Next to this, a bidirectional GRU (Cho et al., 2014) is used for semantic parsing, i.e. mapping the input question to the prefix traversal of the program tree. Importantly, attributes that are used as function arguments in the program are implemented as neural operators. For example, shape and colour are neural operators that map an object embedding onto an embedding of its shape or colour, respectively. Furthermore, the model learns concept embeddings, such as cube or green. Cosine similarity is used to compare the output of the attribute neural operators to the concept embeddings to determine which concept is meant. The functions themselves are implemented symbolically, operating over probability distributions of object embeddings and concept embeddings, which allows the entire model to be differentiable. In sum, the Neuro-Symbolic Concept Learner (NSCL) proposed by Mao et al. (2019) learns object embeddings, concept embeddings, neural operators and the semantic parsing component in an end-to-end manner. No dataset of annotated questions was necessary and the model achieves competitive results (99.2%), even on 10% on the training data (98.9%).

Han et al. (2019) further extend NSCL to the point that meta-questions about the acquired concepts can be answered. In other words, the model does not only learn concepts, but also conceptual categories over these concepts. Specifically, after having learned a number of concept embeddings, one could ask the question if red and green describe the same kind of concept. To achieve this, a meta-operator was added that verifies this question given a pair of concept embeddings. Apart from this, the methodology is identical to NSCL.

### 5.2.3 Discussion

The hybrid procedural semantics methodology is inspired on both modular neural networks approaches and neuro-symbolic approaches. Similar to modular neural network approaches, hybrid procedural semantics combines a program generator (i.e. semantic parsing) with an execution engine. Concretely, computational construction grammar (i.e. the CLEVR grammar from Section 3.4) is used for semantic parsing. This results in a procedural semantic representations consisting of primitive cognitive operators implemented through IRL. The layout of this semantic representation is determined by the grammar’s linguistic analysis of the question. Similar to neuro-symbolic approaches, hybrid procedural semantics flexibly combines the strengths of symbolic and sub-symbolic techniques. Concretely, primitive operators can be implemented either symbolically or sub-symbolically. Again referring to modular approaches, the sub-symbolic primitives are implemented through small, reusable and modular neural networks which have their own task-specific architecture. However, as I will discuss in the next section, hybrid procedural semantics also differs from the aforementioned approaches in several ways in order to accommodate a number of desirable properties.
5.3 Methodology

In hybrid procedural semantics, the meaning underlying an utterance is modelled as a program that can be executed through a combination of symbolic and sub-symbolic primitive cognitive operators. The sub-symbolic primitives are responsible for pattern recognition and are implemented through one or more neural modules, whereas the symbolic primitives can use set operations, arithmetic, search, unification, etc. to accommodate reasoning functionality. Compared to the approaches discussed in Section 5.2, hybrid procedural semantic focusses on a number of properties that allow it to be integrated in human-like communication systems. In what follows, I provide an overview of these properties and how they are operationalised in hybrid procedural semantics.

Compared to modular and neuro-symbolic approaches, hybrid procedural semantics aims to provide a transparent and explainable reasoning process. Specifically, the neural modules supporting sub-symbolic primitives produce either image masks or symbolic labels. Hence, all results produced by neural modules can be readily visualised and inspected. Apart from Mascharka et al. (2018), no other modular neural network approach allows for this. To further open up the black box, neural modules focus on a single, atomic task that cannot be further decomposed, e.g. checking whether an object is a cube or not, corresponding to the third level of specialisation proposed by D’Amario et al. (2021). The specialisation of modules allows to retrace reasoning errors to the behaviour of one specific module, instead of a sub-behaviour of a more generic module. Combined with grammar-based semantic parsing, this ensures that nearly the entire reasoning process is transparent, human-interpretable and explainable.

The specialisation of modules allows the modules to be small in terms of layers and trainable parameters, making hybrid procedural semantics more data-efficient than other approaches. Adding to the data-efficiency is the fact that no questions annotated with semantic representations need to be provided, due to the grammar-based semantic parser.

Hybrid procedural semantics aims to remain open-ended in terms of the concepts it can use. Therefore, neural modules are trained independently from each other which allows new modules, and hence new concepts, to be added more easily. Indeed, new modules can be trained and added without the need to retrain or alter existing modules nor primitive operators. This is not possible when using more generic modules (i.e. specialisation level one or two of D’Amario et al. (2021)) that are jointly trained, as is the case in all modular neural network approaches. I did not make use of the independent training methodology proposed by Castillo-Bolado et al. (2021), as separate training datasets could be generated for each module.

The integration of neural modules in IRL through primitive cognitive operators offers a paradigm that is more powerful than the aforementioned modular and neuro-symbolic approaches. Specifically, all aspects of IRL, as described in Section 2.4, are made available.
even with the integration of neural modules. Most prominently is the implementation of primitive operators as constraints, as opposed to the functional implementation used in most other approaches, with the exception of Manhaeve et al. (2018, 2021). Representing semantics as constraint programs in IRL allows for the execution of semantic networks in multiple directions depending on the data flow, the possibility to return multiple possible solutions and the goal-oriented composition of semantic networks.

5.4 Experimental Setup

The hybrid procedural semantics methodology is demonstrated through a case study on the CLEVR benchmark task. The basis of this case study is the CLEVR grammar (Section 3.4), tackling the benchmark task on the symbolic level. To operationalise hybrid procedural semantics, the primitive cognitive operators used in the procedural semantic representations of the CLEVR grammar need to be altered. I start this section by providing an overview of the primitive cognitive operators and specifically highlight the sub-symbolic primitives and their accompanying neural modules (Section 5.4.1). Afterwards, I focus on the neural network architectures of the modules (Section 5.4.2) and on their training procedure (Section 5.4.3). Finally, I demonstrate how the symbolic and sub-symbolic primitives work together and how they are integrated in IRL to tackle the benchmark task (Section 5.4.4).

5.4.1 Primitive Operators

To operationalise a hybrid procedural semantic representation for the CLEVR benchmark task, I make use of the same set of primitive operators as introduced in Section 3.4.2. This set consists of 14 primitives, each of which model a particular cognitive ability. The primitives can be combined in many different ways to represent the meanings underlying the questions from the CLEVR dataset. While the inner workings of some primitives will remain unchanged, operating on the symbolic level, others will now operate completely on the sub-symbolic level or map sub-symbolic input to symbolic output.

The neural modules that support the sub-symbolic primitives will either perform binary semantic segmentation or classification. Semantic segmentation is the task of predicting the membership to a class on the pixel level, e.g. highlighting and classifying all types of animals in a picture of the savannah. In binary semantic segmentation, the module is trained to recognise just a single class and separate it from the background and other objects, e.g. highlighting only zebras in a picture of the savannah. A binary semantic segmentation module produces an image mask, which is a binary matrix that specifies for every pixel of the input image whether or not the pixel is part of the module’s class. In other words, it is a highlighted area within the original image where instances of the class are found. An image mask can capture zero, one or multiple objects, i.e. a single image mask may contain multiple zebras. However, from the image mask itself, there is
no way of telling how many objects it contains. This needs to be taken into account when
designing the primitives that use binary semantic segmentation modules. Classification
modules take an image mask as input and predict membership to a symbolic class. For
example, a classification module can be used to count the number of objects in the image
mask, predicting a number between zero and ten.

In the following sections, I provide a description of all primitive operators. These are
implemented in the same modes of operation, or directions of processing, as specified in
Section 3.4.2. However, I will focus on the direction of processing that is used in the VQA
task. Finally, I note that all primitives can implicitly access the raw input image without
the need to bind this to one of the arguments.

**Get-Context Primitive**

The (get-context ?context) primitive operator applies binary semantic segmentation to
the image of the CLEVR scene and produces an image mask containing all objects in the
scene. The resulting mask is bound to the variable ?context. The operator is illustrated in
Figure 5.1. This primitive is used in the semantic representation of every question. Other
sub-symbolic primitives operate over the image mask produced by get-context.

**Filter Primitive**

The (filter ?output-mask ?input-mask ?concept) primitive filters the mask of a set of
objects (?input-mask) on the basis of a ?concept, such as blue or cube. First, it uses a
semantic segmentation module to create an image mask containing all instances of ?concept
from the raw input image. These modules are called find modules. Multiple highly
specific find modules are available, namely one for each concept. Depending on the bind-
ing value of ?concept, the appropriate module is selected. Afterwards, a mask intersection
operation is applied to the ?input-mask and the image mask produced by the find mod-
ule. This allows for consecutive filter primitives, as the ?input-mask may already be a
filtered subset of the objects in the scene. The resulting image mask is bound to ?output-
mask. This primitive operator is illustrated in Figure 5.2, where a filter[blue] operator is
fed with the output of a filter[cylinder] operator, resulting in the set of blue cylinders
from the input image (in this case only one).

**Query Primitive**

The (query ?concept ?object-mask ?category) primitive operator queries the ?cate-
gory, such as colour or material, of a particular ?object-mask. The object mask is first
multiplied with the input image and a classification module is used to query the ?cate-
gory of the highlighted object, returning a particular ?concept such as green or metal.
An example query operator is shown in Figure 5.3. In practice, the query primitive uses
CHAPTER 5. LEARNING CONCEPTS AS NEURAL MODULES

Figure 5.1: Schematic representation of the GET-CONTEXT primitive operator.

Figure 5.2: Schematic representation of the FILTER primitive operator.
a number of highly specific binary classification modules, each of which make a yes-or-no decision. For instance, when querying the material of an object as in Figure 5.3, a `query[metal]` and a `query[rubber]` module are used. The module with the highest probability for ‘yes’ wins and the associated concept is returned.

**SAME Primitive**

The goal of the (same ?output-mask ?object-mask ?category) primitive is to compute the set of objects (?output-mask) that has the same value for the ?category as the category of a given source object (?object-mask), excluding the source object itself. For example, it is used to find the set of all objects that has the same shape as a given object. Internally, the same primitive first uses a query operator to access the concept of the source object belonging to the given category, followed by a find module to find all objects in the input image which possess that concept. Finally, the ?object-mask is subtracted from the mask computed by the find module to exclude the source object. Figure 5.4 illustrates this process. This primitive is a clear example of the modularity of hybrid procedural semantics, allowing to reuse modules across multiple primitive operators. As before, highly specific find modules and query modules are used. Specifically, the example illustrated in Figure 5.4 uses `query[cube]`, `query[sphere]`, `query[cylinder]` and `find[cube].`

**Relate Primitive**

The (relate ?output-mask ?object-mask ?spatial-relation) operator computes all objects for which the ?spatial-relation holds with respect to a source object (?object-mask). For example, the primitive can be used to compute all objects that are left of a given object. The resulting image mask is bound to ?output-mask. This is achieved by a semantic segmentation module that is fed with the ?object-mask and the input image. Multiple highly specific binary semantic segmentation models are available, namely one for every spatial relation. The correct one is selected on the basis of ?spatial-relation. An example for relate[left] is provided in Figure 5.5.

**COUNT Primitive**

The (count ?number ?input-mask) primitive computes the number of objects in the ?input-mask. Specifically, a classification module is fed with the ?input-mask multiplied with the input image and predicts a number between zero and ten. The predicted number is bound to ?number. Figure 5.6 provides an example.

---

1 Scenes in the CLEVR dataset contain maximally ten objects.
CHAPTER 5. LEARNING CONCEPTS AS NEURAL MODULES

Figure 5.3: Schematic representation of the query primitive operator.

Figure 5.4: Schematic representation of the same primitive operator.
5.4. EXPERIMENTAL SETUP

Figure 5.5: Schematic representation of the RELATE primitive operator.

Figure 5.6: Schematic representation of the COUNT primitive operator.
EXIST Primitive

The (exist ?BOOLEAN ?INPUT-MASK) primitive operator is used to check whether the set of objects, represented through ?INPUT-MASK, is empty or not. Internally, it makes use of the count classification module. If this returns zero, ?BOOLEAN is bound to NO. In all other cases, it is bound to YES. This primitive is illustrated in Figure 5.7.

UNIQUE Primitive

The (unique ?OBJECT-MASK ?INPUT-MASK) primitive operator will check if ?INPUT-MASK contains a single object. If so, ?OBJECT-MASK is bound to the same image mask as ?INPUT-MASK. Otherwise, ?OBJECT-MASK remains unbound, causing the primitive operator to fail. Internally, the count classification module is used, as shown in Figure 5.8.

INTERSECT Primitive

The (intersect ?OUTPUT-MASK ?INPUT-MASK-1 ?INPUT-MASK-2) primitive operators computes the intersection of both input masks. As these are binary matrices, an element-wise min operation is used. The result is bound to ?OUTPUT-MASK. Figure 5.9 provides a schematic visualisation.

UNION Primitive

The (union ?OUTPUT-MASK ?INPUT-MASK-1 ?INPUT-MASK-2) primitive operators computes the union of both input masks. As these are binary matrices, an element-wise max operation is used. The result is bound to ?OUTPUT-MASK. Figure 5.10 provides a schematic visualisation.

EQUAL Primitive

The (equal ?BOOLEAN ?CONCEPT-1 ?CONCEPT-2 ?CATEGORY) primitive operates on the symbolic level. It receives two concepts of the same ?CATEGORY, e.g. ?CONCEPT-1 being METAL and ?CONCEPT-2 being RUBBER, both of the MATERIAL category, and checks whether these are equal. The ?BOOLEAN is bound to YES or NO.

EQUAL-INTEGER / LESS-THAN / GREATER-THAN Primitives

The primitives (equal-integer ?BOOLEAN ?NUMBER-1 ?NUMBER-2), (less-than ?BOOLEAN ?NUMBER-1 ?NUMBER-2) and (greater-than ?BOOLEAN ?NUMBER-1 ?NUMBER-2) symbolically compute whether the numbers bound to ?NUMBER-1 and ?NUMBER-2 are respectively equal, less than or greater than one another.

Table 5.2 provides an overview of all primitive operators with respect to their input and output data types. The primitives can be categorised in three groups: (i) mapping sub-symbolic input to sub-symbolic output, (ii) mapping sub-symbolic input to symbolic output.
5.4. EXPERIMENTAL SETUP

Figure 5.7: Schematic representation of the \texttt{exist} primitive operator.

Figure 5.8: Schematic representation of the \texttt{unique} primitive operator.

Figure 5.9: Schematic representation of the \texttt{intersect} primitive operator.

Figure 5.10: Schematic representation of the \texttt{union} primitive operator.
and (iii) mapping symbolic input to symbolic output. The first group of primitives operates over the images and image masks. The second group of primitives maps these image masks to symbolic data. The third group of primitives reasons over the symbolic data. As can be seen from this table, there are no primitives mapping symbolic input to sub-symbolic output. The hybrid procedural semantics approach for CLEVR thus consists of a pipeline in terms of the data types, passing along information from the sub-symbolic level to the symbolic level.

Table 5.2: Overview of primitive operators categorised by the symbolic or sub-symbolic nature of their input and output arguments.

<table>
<thead>
<tr>
<th>Input</th>
<th>symbolic</th>
<th>sub-symbolic</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbolic</td>
<td>EQUAL</td>
<td>QUERY</td>
</tr>
<tr>
<td></td>
<td>EQUAL-INTEGER</td>
<td>COUNT</td>
</tr>
<tr>
<td></td>
<td>LESS-THAN</td>
<td>EXIST</td>
</tr>
<tr>
<td></td>
<td>GREATER-THAN</td>
<td></td>
</tr>
<tr>
<td>sub-symbolic</td>
<td>GET-CONTEXT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILTER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIQUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RELATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTERSECT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNION</td>
<td></td>
</tr>
</tbody>
</table>

5.4.2 Module Architectures

Having provided an overview of the primitive operators, I now discuss the architectures of the neural modules that are used to implement them. Specifically, two types of modules are used across the sub-symbolic primitive operators: binary semantic segmentation modules and classification modules. The former rely on the U-Net architecture (Ronneberger et al., 2015) and the latter on the SqueezeNet architecture (Iandola et al., 2016). In what follows, I discuss both of these architectures and provide a detailed specification of the neural modules. All neural modules were implemented using the PyTorch framework (Paszke et al., 2019).

Semantic Segmentation Modules

The U-Net architecture by Ronneberger et al. (2015) was specifically designed for the biomedical domain. In that domain, it is not only important to correctly classify (parts of) images, e.g. cells being malignant or benign, but also to precisely localize these classes in the imaging, e.g. which cells are malignant or benign. This is exactly the task of semantic segmentation. Moreover, a data-efficient neural architecture was sought after, since it
is difficult to obtain large training corpora of annotated imaging.

Figure 5.11 provides a schematic overview of the U-Net architecture. It consists of a contracting path and an expansive path, giving it the U-shaped network layout. These paths are made up of a number of 'blocks’. A single block consists of two 3x3 convolutional layers, each followed by a ReLU activation function. The number of blocks that is used can be adjusted according to the complexity of the task.

![Schematic representation of the U-Net architecture from Ronneberger et al. (2015)](image)

The contracting path reduces the dimensions of the input image and thereby captures contextual information (Ronneberger et al., 2015). Specifically, after each block, a 2x2 max pooling operation is used for reducing the dimensions of the feature map while doubling its number of channels.

The expansive path again increases the number of dimensions and halves the number of channels, providing precise localization (Ronneberger et al., 2015). On this path, blocks are connected by upsampling layers through a 2x2 bilinear interpolation operation that is applied to the feature map. This operation uses neighbouring pixels to compute a pixel’s value in the upsampled feature map. Blocks on the expansive path also make use of ‘skip connec-
tions’ that concatenate the feature map from the corresponding depth on the contracting path to the feature map on the expansive path. The feature map from the contracting path is cropped to account for border pixels that got lost by the unpadded convolutions.

The final layer consists of 1x1 convolutions and a sigmoid activation function to produce the desired number of channels, namely one for each class. Each channel specifies a ‘soft attention’ for a class, i.e. a matrix that has roughly the same size as the input image where every pixel has a value between 0 and 1, denoting the probability that the pixel belongs to the class. This is easily transformed into a single-channel mask by providing every pixel with the label of the class that has the highest probability.

In the following paragraphs, detailed specifications of the neural modules are provided detailing the modifications made to the original U-Net architecture.

Blocks. The neural modules use a modified ‘block’ layout, as compared to Ronneberger et al. (2015). Specifically, a batch normalization layer is added in between the 3x3 convolutional layer and the ReLU activation function. This has become standard practice in recent years as it allows for faster and more stable training of deep networks (Ioffe and Szegedy, 2015). Arguably, it was not yet added to the U-Net architecture because it was introduced in the same year.

Get-Context Module. Due to the relative simplicity of its task, the get-context module uses only two blocks in both the contracting and the expansive path.

Find Modules. Each find module consists of six blocks on both the contracting and the expansive path. Except for the first block, the contracting path uses halve blocks, i.e. a single 3x3 convolutional layer, batch normalisation layer and ReLU activation function.

Relate Modules. Each relate module first combines the input image with an image mask produced by another module. This is done by applying a ‘block’ to each input and multiplying the results. Afterwards, as in the find module, six halve blocks are used in the contracting path and six full blocks are used in the expansive path.

Classification Modules

The SqueezeNet architecture (Iandola et al., 2016) was specifically designed to achieve competitive levels of accuracy on image classification tasks with much fewer trainable parameters. This has the advantage of (i) more efficient training on distributed machines, as fewer data needs to be communicated, (ii) smaller file-sizes for trained models, allowing for easier distribution and (iii) smaller memory requirements, allowing to use the model
The major breakthrough in this architecture is the introduction of the ‘Fire module’, illustrated in Figure 5.12. A Fire module consists of a squeeze layer, comprised of 1x1 convolutional filters, followed by an expand layer, comprised of a mixture of 1x1 and 3x3 convolutional filters. ReLU activation functions are added after each layer. The design of the Fire module helps to reduce the number of trainable parameters (Iandola et al., 2016). Figure 5.13 illustrates the complete SqueezeNet architecture. Downsampling operations are placed relatively late in the network architecture. This allows the convolutions to operate on larger feature maps with the aim of maximizing the accuracy on a limited budget of parameters, due to the Fire modules.

Figure 5.13: Schematic overview of the SqueezeNet architecture. Image adapted from Iandola et al. (2016).

In the following paragraphs, detailed specifications of the neural modules are provided detailing the modifications made to the original SqueezeNet architecture.

---

2Ronneberger et al. (2015) use unpadded convolutions, resulting in a fixed-width border that gets lost through the contracting and expansive path.
QUERY Modules. As discussed in Section 5.4.1, several highly specific QUERY modules are used, each trained to make a yes-or-no decision about the presence or absence of a particular concept in an object. This requires two modifications to the SqueezeNet architecture. First, the input of the QUERY modules consists of the element-wise multiplication of the input image and an image mask. Second, the last convolutional layer is changed such that it outputs a feature map with two channels.

COUNT Module. The COUNT module is trained to predict integers between zero and ten. Therefore, the last convolutional layer of the SqueezeNet architecture produces a feature map of 11 channels. The input to the COUNT module consists of an input image and an image mask. Both are passed through a 3x3 convolutional layer, batch normalization layer, ReLU activation function and 2x2 max pooling operation. The element-wise multiplication of these results is passed as input to the SqueezeNet architecture.

5.4.3 Training

To maximize the open-endedness of the hybrid procedural semantics approach, the neural modules are trained independently from each other. This allows to extend the library of modules at any time. For example, when a new shape is introduced in the dataset, specialized FIND and QUERY modules can be trained to capture this concept, after which they are added to the corresponding primitive operations without the need to retrain or adjust any existing modules. Additionally, it allows to evaluate the neural modules independently from each other. This in contrast to other modular neural networks approaches where modules can only be trained and evaluated in an end-to-end fashion. As a consequence of the independent training procedure, separate training, validation and test sets for every neural module needed to be generated. In the following sections, I describe this data generation process and I provide an overview of the modules’ hyper-parameters that were used during training. In total, 36 modules were trained using HPC infrastructure provided by VSC (Vlaams Supercomputer Centrum) on modern CPU (Intel Xeon) and GPU (Nvidia Tesla P100, Nvidia A100, Nvidia Volta V100) platforms.

Dataset Generation

To generate datasets for each of the modules, annotated samples needed to be provided. For semantic segmentation modules, the inputs are images (GET-CONTEXT and FIND) or image masks (RELATE) and the outputs are image masks. For classification modules, the inputs are image masks and the outputs are labels. The main challenge in generating these datasets thus consists in constructing image masks that contain only the desired objects.

To overcome this challenge, a Mask R-CNN model (He et al., 2017) pre-trained by Yi et al. (2018) to detect and segment CLEVR objects was used. This model provides a separate image mask for every object in the image. These masks are matched with the symbolic an-
notations of the scene provided by the CLEVR dataset (see Section 3.3), such that for every object in the scene there is both a complete symbolic description of its properties and a corresponding image mask containing only that object. The matching operation computes the Euclidean distance between the symbolically specified coordinates and the coordinates of the segmented objects. Each mask is assigned to the nearest object. With this combined information, it becomes straightforward to generate the desired image masks. In the following paragraphs, the data generation process for each module is illustrated through an example.

**Get-Context Module.** This module is trained to detect and segment all objects in the scene. Output masks are obtained by combining all object masks of a scene into a single mask. Thus, a single sample can be generated for every CLEVR image.

**Find Module.** A find[cube] module takes the image as input and should produce an output that is an image mask containing only cubes. Using the symbolic annotation, all cubes from the scene are selected and their respective image masks are combined into one. With this method, a single sample can be generated for every image.

**Query Module.** A query[rubber] module requires an input that is an image mask containing just a single object and an output that is either 'yes' or 'no'. The input is provided by choosing a random object from the scene and taking the corresponding mask. The symbolic annotation of that object is used to determine the correct output label. Samples for every object in every image can be generated as such.

**Relate Module.** The relate[left] module takes a mask with a single object as input. The output masks are generated by combining all image masks of the objects that are left of the input object. This is determined using the symbolic annotation. Such samples can be generated for every object in every image.

**Count Module.** The count module takes an image mask as input and a symbolic label, denoting a number, as output. Using the combined information, an image mask can be generated for every subset of objects by combining their masks into one. The output label is added by taking the cardinality of the set of objects on the symbolic annotation level.

Table 5.3 provides an overview of the number of samples that could be generated for each module. Since the data generation process relies on the symbolic scene annotations, CLEVR’s original test split could not be used because the symbolic annotations of this split is not released. Therefore, 10,000 images from the training set were kept separately and used as a test set, thereby reducing the training set from 70,000 to 60,000 images. CLEVR’s validation set, containing 15,000 images, was kept as is. For efficiency reasons, the datasets for the count module were reduced in size. The number of samples that were
Table 5.3: Overview of the number of samples per neural module. The notation ‘find[*]’ refers to the set of all FIND modules. The number of FIND modules is added between parentheses.

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Training Set</th>
<th>Validation Set</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>60,000</td>
<td>15,000</td>
<td>10,000</td>
</tr>
<tr>
<td>find[*] (15)</td>
<td>60,000</td>
<td>15,000</td>
<td>10,000</td>
</tr>
<tr>
<td>relate[*] (4)</td>
<td>390,805</td>
<td>97,358</td>
<td>64,827</td>
</tr>
<tr>
<td>query[*] (15)</td>
<td>390,805</td>
<td>97,358</td>
<td>64,827</td>
</tr>
<tr>
<td>count</td>
<td>15,453,136</td>
<td>3,813,376</td>
<td>2,542,688</td>
</tr>
</tbody>
</table>

used for training and evaluating this module is noted between parentheses in Table 5.3. These samples were chosen at random.

Hyper-parameters

Table 5.4 provides an overview of the hyper-parameters of each module. BCE and NLL, used in the ‘Loss Function’ column, refer to binary cross entropy loss and negative log likelihood loss, respectively. The evaluation functions IOU and MC refer to intersection over union and multi-class evaluation respectively. The former is used to evaluate the semantic segmentation modules. It divides the overlap (intersection) of the predicted segment and the ground-truth segment by the union of those segments. This returns a value between 0 and 1, with 1 corresponding to perfectly overlapping segments. The latter is used for the classification modules and simply counts the number of correctly predicted labels. All modules were trained using the Adam optimizer (Kingma and Ba, 2014).

Table 5.4: Overview of hyper-parameters used for neural modules. The notation ‘find[*]’ refers to the set of all FIND modules. The number of FIND modules is added between parentheses.

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Learning Rate</th>
<th>Weight Decay</th>
<th>Batch Size</th>
<th>Time-steps</th>
<th>Loss Function</th>
<th>Evaluation Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>0.001</td>
<td>0</td>
<td>128</td>
<td>20,000</td>
<td>BCE</td>
<td>IOU</td>
</tr>
<tr>
<td>find[*] (15)</td>
<td>0.001</td>
<td>0</td>
<td>128</td>
<td>20,000</td>
<td>BCE</td>
<td>IOU</td>
</tr>
<tr>
<td>relate[*] (4)</td>
<td>0.001</td>
<td>0</td>
<td>128</td>
<td>20,000</td>
<td>BCE</td>
<td>IOU</td>
</tr>
<tr>
<td>query[*] (15)</td>
<td>0.0001</td>
<td>0.0001</td>
<td>128</td>
<td>50,000</td>
<td>NLL</td>
<td>MC</td>
</tr>
<tr>
<td>count</td>
<td>0.001</td>
<td>0</td>
<td>128</td>
<td>65,000</td>
<td>NLL</td>
<td>MC</td>
</tr>
</tbody>
</table>

5.4.4 Neural Modules Interface

Before moving on to the experimental results, I discuss how the primitive operators, implemented through IRL, interface with the neural modules, implemented in Python through
the PyTorch framework. Specifically, this is operationalised using a client-server architecture, where the primitive operators in IRL act as clients, while the neural modules are accessible through a Python server. Communication is handled over HTTP.

On the server side, a web service is set up using the Flask framework. Upon start-up, the web service loads the trained neural modules in memory and makes them centrally available. Inspired by RESTful software architectures, resources on this Flask web service are identified through their URI. Specifically, one URI, or endpoint, is provided for every primitive operator. For example, when needing to execute the filter primitive operator, an HTTP request is sent to `http://<server-address>/filter`.

In IRL, the primitive operators are implemented such that they make calls to their respective endpoints over HTTP. Concretely, these are HTTP POST requests that specify the arguments of the primitive operator and their values. A null value is provided for unbound arguments. Depending on the bound and unbound arguments, the web service can figure out how it should apply the neural module(s) associated to the endpoint that received the HTTP POST request. For instance, if the arguments `?input-mask` and `?concept` of the filter primitive are bound, the find module that is associated to that `?concept` is retrieved, e.g. `find[cube]` and applied to the `?input-mask`. Alternatively, if the `?input-mask` and the `?output-mask` are provided, all find modules are applied to the `?input-mask` and the concept of the find module that maximises the IOU evaluation function over the resulting mask and the provided `?output-mask` is determined. The HTTP responses consist of new bindings, i.e. one or multiple values for every unbound argument. This set-up ensures that the multidirectionality of primitive operators in IRL remains possible also in hybrid procedural semantics. Synchronous communication is used, and requests and responses are encoded in the JSON format.

The communication scheme described above results in many HTTP requests, namely one request for every sub-symbolically implemented primitive operator that is executed. To make these requests and responses more lightweight, attentions that are computed by the neural modules are not transmitted over HTTP. Instead, they are stored centrally on the server and each given a unique identifier. IRL simply operates using these identifiers. When such an identifier is used in a subsequent HTTP request, the server can look up the corresponding attention and feed it as input to the correct neural module. For visualisation and interpretability purposes, an endpoint is made available through which IRL can request an attention with a given identifier. Concretely, to retrieve the attention with identifier `attn-1`, a HTTP GET request is sent to `http://<server-address>/attn/attn-1`. Using this endpoint, the execution of a semantic network and all masks that were computed during this process can be visualised in Babel’s web interface.

A third and final endpoint is provided for loading the image associated to the CLEVR scene in which the semantic network should be executed. Concretely, IRL sends a HTTP POST request specifying the unique name of the CLEVR scene to the web service. In turn,
symbolically in IRL through their unique identifiers. HTTP over HTTP is a Flask web service that exposes an endpoint every neural module. Attention is stored on the server and represented in IRL as neurons and neural modules. Primitive operators send requests to the service in the form of HTTP POST requests.

**Figure 5.14**: Schematic overview of the integration of neural modules in IRL (read from right to left). Primitive operators send requests to the Flask web service, which processes them and returns responses.
5.5. EXPERIMENTAL RESULTS

the web service loads the image and makes it centrally available such that subsequent primitive operators can access it. Additionally, the web service will clear all attentions that it computed during previous executions of semantic networks, as these are no longer required. This way, the memory of the web service does not overflow.

Using this architecture, the evaluation of the CLEVR task through hybrid procedural semantics proceeds as follows. The CLEVR grammar (Section 3.4) maps the natural language question onto its underlying meaning representation in the form of a semantic network. Before evaluating this semantic network, the CLEVR image is loaded in memory on the server side. The primitive operators of the network are executed in a hybrid way, combining the strengths of symbolic and sub-symbolic techniques. For the execution of sub-symbolic primitives, HTTP requests are sent to the Python web service running the neural modules. This is illustrated in Figure 5.14. The output of each neural module, in the form of an attention, can be downloaded on request. This happens automatically when visualising the execution of semantic networks in Babel’s web interface. Crucially, the integration of computational construction grammar and hybrid procedural semantics allows most of the question answering process to be transparent, explainable and human-interpretable while operating directly on raw image data.

5.5 Experimental Results

In this section, I present the experimental results of the hybrid procedural semantics approach. First, the neural modules described in Section 5.4 are evaluated independently from each other. These results are discussed in Section 5.5.1. Afterwards, evaluation is carried out through the CLEVR benchmark task in Section 5.5.2. Finally, in Section 5.5.3, I test the systematic generalisation abilities of the neural modules. Throughout this section, I compare hybrid procedural semantics both to the state-of-the-art models discussed in Section 5.2 and to the grounded concept learning approach discussed in Chapter 4.

5.5.1 Neural Module Evaluation

The neural modules are first evaluated on their respective held-out test sets, described in Table 5.3. Both the loss and the accuracy are reported in Table 5.5. This table lists the semantic segmentation modules on the left side and the classification modules on the right side. As can be seen from Table 5.5, all modules perform extremely well and consistently achieve over 99% accuracy, with 15 out of 36 modules obtaining over 99.9% accuracy.

The performance achieved by the individual neural modules is comparable to that of the discrimination-based concepts learned in the simulated environment (Section 4.5.1), even though the neural modules operate directly on the images. However, a number of key differences should be noted. First, although the neural modules are designed to capture atomic tasks in order to enhance the overall interpretability, the reasoning process that is
Table 5.5: Evaluation results of each neural module on the held-out test set.

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Loss (BCE)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-context</td>
<td>0.010</td>
<td>99.20</td>
</tr>
<tr>
<td>find[blue]</td>
<td>0.002</td>
<td>99.91</td>
</tr>
<tr>
<td>find[brown]</td>
<td>0.002</td>
<td>99.90</td>
</tr>
<tr>
<td>find[cube]</td>
<td>0.005</td>
<td>99.67</td>
</tr>
<tr>
<td>find[cyan]</td>
<td>0.002</td>
<td>99.90</td>
</tr>
<tr>
<td>find[cylinder]</td>
<td>0.004</td>
<td>99.74</td>
</tr>
<tr>
<td>find[gray]</td>
<td>0.002</td>
<td>99.89</td>
</tr>
<tr>
<td>find[green]</td>
<td>0.002</td>
<td>99.90</td>
</tr>
<tr>
<td>find[large]</td>
<td>0.008</td>
<td>99.52</td>
</tr>
<tr>
<td>find[metal]</td>
<td>0.006</td>
<td>99.64</td>
</tr>
<tr>
<td>find[purple]</td>
<td>0.002</td>
<td>99.90</td>
</tr>
<tr>
<td>find[red]</td>
<td>0.002</td>
<td>99.90</td>
</tr>
<tr>
<td>find[rubber]</td>
<td>0.007</td>
<td>99.63</td>
</tr>
<tr>
<td>find[small]</td>
<td>0.007</td>
<td>99.74</td>
</tr>
<tr>
<td>find[sphere]</td>
<td>0.003</td>
<td>99.83</td>
</tr>
<tr>
<td>find[yellow]</td>
<td>0.002</td>
<td>99.89</td>
</tr>
<tr>
<td>relate[behind]</td>
<td>0.005</td>
<td>99.64</td>
</tr>
<tr>
<td>relate[front]</td>
<td>0.006</td>
<td>99.55</td>
</tr>
<tr>
<td>relate[left]</td>
<td>0.005</td>
<td>99.65</td>
</tr>
<tr>
<td>relate[right]</td>
<td>0.004</td>
<td>99.66</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Loss (NLL)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>0.051</td>
<td>99.05</td>
</tr>
<tr>
<td>query[blue]</td>
<td>0.003</td>
<td>99.97</td>
</tr>
<tr>
<td>query[brown]</td>
<td>0.004</td>
<td>99.95</td>
</tr>
<tr>
<td>query[cube]</td>
<td>0.010</td>
<td>99.79</td>
</tr>
<tr>
<td>query[cyan]</td>
<td>0.003</td>
<td>99.97</td>
</tr>
<tr>
<td>query[cylinder]</td>
<td>0.010</td>
<td>99.80</td>
</tr>
<tr>
<td>query[gray]</td>
<td>0.004</td>
<td>99.95</td>
</tr>
<tr>
<td>query[green]</td>
<td>0.005</td>
<td>99.94</td>
</tr>
<tr>
<td>query[large]</td>
<td>0.012</td>
<td>99.82</td>
</tr>
<tr>
<td>query[metal]</td>
<td>0.008</td>
<td>99.87</td>
</tr>
<tr>
<td>query[purple]</td>
<td>0.005</td>
<td>99.94</td>
</tr>
<tr>
<td>query[red]</td>
<td>0.004</td>
<td>99.95</td>
</tr>
<tr>
<td>query[rubber]</td>
<td>0.008</td>
<td>99.87</td>
</tr>
<tr>
<td>query[small]</td>
<td>0.012</td>
<td>99.81</td>
</tr>
<tr>
<td>query[sphere]</td>
<td>0.006</td>
<td>99.91</td>
</tr>
<tr>
<td>query[yellow]</td>
<td>0.005</td>
<td>99.95</td>
</tr>
</tbody>
</table>
5.5. EXPERIMENTAL RESULTS

internal to each module is still a black box. There is no way of knowing what features were used to decide, for example, whether a particular object is of a given shape or not, in contrast to the grounded concepts which explicitly list those features and their prototypical values. Second, the neural modules allow for internal inconsistencies as separate modules are required to capture one particular concept. Specifically, one module is required to identify objects with a certain attribute (i.e. FIND modules) while another is required to identify an attribute of a certain object (i.e. QUERY modules). These modules do not rely on the same internal concept representation, which allows these related modules to contradict each other. Third, even though the neural modules are data-efficient compared to other neural approaches, much more data and training time was needed when compared to the grounded concept learning approach. Specifically, neural modules require multiple ‘epochs’ of training, which consists of revisiting the entire training dataset multiple times. In contrast, the grounded concepts could be used successfully in communication after ~1000 interactions.

5.5.2 CLEVR Benchmark

After evaluating the neural modules independently, I present the evaluation results of the complete hybrid procedural semantics approach applied to the CLEVR benchmark (Johnson et al., 2017a). This consists in mapping each natural language question onto a semantic network and executing the semantic network in a hybrid way, as described in Section 5.4.4. The evaluation is carried out using the questions that accompany the 10,000 images that were held out as a test set. Since ten questions are available for every image in the CLEVR dataset, the test set consists of 100,000 questions.

Table 5.6 provides an overview of the results. The question answering accuracy is split per ‘question type’, which corresponds to the primitive operator that is used to compute the final answer. The last column in Table 5.6 provides the overall accuracy, i.e. the percentage of correctly answered questions, which is at 99.2%. The question answering accuracy is slightly lower than the accuracy of the individual modules as mistakes made by individual modules may propagate and accumulate through the semantic network. Table 5.6 also compares hybrid procedural semantics against several state-of-the-art models from Section 5.2. Hybrid procedural semantics achieves results that are directly comparable to the state-of-the-art modular neural network approach (99.1% by Mascharka et al. (2018)) and the state-of-the-art neuro-symbolic approach (99.8% by Yi et al. (2018)). The analysis per question type reveals that questions requiring the COUNT primitive operators are most difficult. This can also be observed for all of the other models in Table 5.6.

Similar to the module-specific evaluation, the neural modules outperform the grounded concepts in terms of the CLEVR task. Using the grounded concepts, a question answering accuracy of 96.2% in the simulated environment could be achieved (see Section 4.6), whereas the neural modules nearly solve the task. However, as discussed in Section 5.5.1,
Table 5.6: Performance of hybrid procedural semantics on the CLEVR VQA task.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare Integer</td>
<td>NMN (Andreas et al., 2016a)</td>
<td>79.3</td>
<td>52.5</td>
<td>72.5</td>
<td>79.0</td>
<td>78.0</td>
<td>72.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEP (Johnson et al., 2017b)</td>
<td>97.1</td>
<td>92.7</td>
<td>98.0</td>
<td>99.0</td>
<td>98.9</td>
<td>98.8</td>
<td>98.4</td>
<td>98.1</td>
<td>97.3</td>
<td>99.8</td>
<td>98.5</td>
</tr>
<tr>
<td></td>
<td>N2NMN (Hu et al., 2017a)</td>
<td>85.7</td>
<td>68.5</td>
<td>73.8</td>
<td>88.4</td>
<td>96.1</td>
<td>91.5</td>
<td>94.3</td>
<td>91.5</td>
<td>90.6</td>
<td>92.6</td>
<td>82.8</td>
</tr>
<tr>
<td></td>
<td>MAC (Hudson and Manning, 2018)</td>
<td>99.5</td>
<td>97.2</td>
<td>99.4</td>
<td>99.3</td>
<td>99.5</td>
<td>98.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TBQ (Kashima, 2018)</td>
<td>99.2</td>
<td>97.6</td>
<td>99.4</td>
<td>99.5</td>
<td>99.6</td>
<td>99.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M2N-VQA (Hudson et al., 2017b)</td>
<td>99.9</td>
<td>99.7</td>
<td>99.9</td>
<td>99.8</td>
<td>99.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NSCL (Mao et al., 2019)</td>
<td>98.8</td>
<td>98.2</td>
<td>99.0</td>
<td>99.3</td>
<td>99.1</td>
<td>98.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hybrid Procedural Semantics</td>
<td>99.6</td>
<td>98.0</td>
<td>99.2</td>
<td>99.6</td>
<td>99.4</td>
<td>99.5</td>
<td>99.7</td>
<td>99.7</td>
<td>99.3</td>
<td>99.6</td>
<td>99.7</td>
</tr>
</tbody>
</table>

Note: Andreas et al. (2016a) did not evaluate their model on the CLEVR dataset, but this evaluation was carried out and reported by Mao et al. (2019).
the major advantages of the grounded concepts include their complete transparency and interpretability, the internal consistency and the data-efficiency.

### 5.5.3 Generalisation Experiment

The third and final experiment tests the ability of hybrid procedural semantics to generalise to similar, yet unseen, combinations of visual features. The experiment is operationalised through the CLEVR CoGenT dataset, previously introduced in Section 3.3.4. The CoGenT dataset consists of two experimental conditions where one set of feature combinations is available for training and another set of feature combinations is available for evaluation. Concretely, cubes and cylinders each have four possible colours in condition A and these colour options are switched in condition B. Spheres can have any of the eight available colours in both experimental conditions. The neural modules are trained on the images of condition A, using the same neural architectures, training procedure and hyperparameters as described in Section 5.4.3. Afterwards, they are evaluated on condition B, both using a held-out module-specific test set and through the visual question answering task. The goal of the experiment is to investigate how well the neural modules truly capture the concept they are trained for. For instance, will the filter[cube] module recognise red cubes in condition B if it was only trained on differently coloured cubes in condition A? Ideally, one would want that a filter[cube] module is not "distracted" by the colour of the cubes it is trained on and that it recognises cubes through shape-related features, as was shown for the discrimination-based concepts in Section 4.5.2.

The evaluation results on condition A of the CoGenT dataset are equivalent with those reported in Sections 5.5.1 and 5.5.2. Namely, all neural modules achieve over 99% accuracy on a held-out test set and perform equally well on the VQA task, namely 99.4% question answering accuracy. For space reasons, the exact numbers are reported in the supplementary materials accompanying this chapter (Appendix B, Table B.1).

On condition B, the specific neural modules that are listed in Table 5.7 show a decrease in performance on the held-out test set. These results are reported on the left side of Table 5.7, under ‘Before FT’. Similarly, the overall question answering accuracy decreases from 99.4% to 70.7%. For space reasons, the loss and accuracy of all neural modules and the question answering accuracy per question type on condition B are provided in Appendix B, Table B.2 and Table B.3, respectively. These results indicate that the neural modules responsible for finding and recognising cubes and cylinders have not truly learned the underlying concepts. These are exactly the shapes for which the colour options are switched in between condition A and B. Instead, these neural modules have learned to find and recognise these shapes (partly) on the basis of their colour despite the fact that several colour options were available for each shape.

These results stand in stark contrast with the results obtained in Section 4.5.2. The agents using the discriminated-based concept representation did not suffer any decrease in com-
municative success when transitioning from condition A to B (see Figure 4.7). Specifically, the various colour options that were available for cubes and cylinders in condition A were sufficient for the agent to learn that the shape-related features are indeed discriminative.

Table 5.7: Loss and accuracy of selected neural modules on both conditions A and B before and after finetuning (FT) on condition B.

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Before FT</th>
<th></th>
<th></th>
<th>After FT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>find[cube]</td>
<td>0.004</td>
<td>99.78</td>
<td>1.355</td>
<td>81.58</td>
<td>1.417</td>
<td>82.02</td>
</tr>
<tr>
<td>find[cylinder]</td>
<td>0.003</td>
<td>99.80</td>
<td>0.872</td>
<td>81.52</td>
<td>0.337</td>
<td>82.60</td>
</tr>
<tr>
<td>query[cube]</td>
<td>0.011</td>
<td>99.86</td>
<td>5.417</td>
<td>33.19</td>
<td>5.934</td>
<td>38.40</td>
</tr>
<tr>
<td>query[cylinder]</td>
<td>0.007</td>
<td>99.90</td>
<td>5.038</td>
<td>35.77</td>
<td>3.447</td>
<td>66.68</td>
</tr>
</tbody>
</table>

Following common practice in the neural network literature, ‘finetuning’ may be applied to the neural modules. Finetuning is a technique for using a neural network on a different (yet similar) dataset than the one it was trained on without completely retraining the network. It consists of providing additional training examples from the new dataset using a fraction of the samples compared the original training data. The motivation for finetuning is that only a few samples should be sufficient to steer the neural network’s weights in “the right direction” such that it can be used on the new dataset. In this case, samples from condition B are provided with the aim of steering the neural modules trained on condition A such that they generalise over both experimental conditions of the CoGenT dataset. Following Johnson et al. (2017b) and others, 3,000 images from condition B were used for finetuning the neural modules listed in Table 5.7. These modules were not trained until convergence on condition B, but only finetuned for a few epochs. Crucially, thanks to the independent design and training of the neural modules in hybrid procedural semantics, these modules could be finetuned separately. All other modules could be left as is and perform equally well on conditions A and B (see Appendix B). In other work, this is typically not possible as all modules are trained in an end-to-end fashion and thus also need to be finetuned as such.

The loss and accuracy of the finetuned neural modules on the held-out test sets of conditions A and B are reported on the right side Table 5.7, under ’After FT’ where FT stands for finetuning. Generally speaking, the finetuned neural modules show a decrease in performance on the data on which they were originally trained (condition A) and an increase in performance on the data on which they are finetuned (condition B). In other words, these modules seem to have traded their performance on condition A for performance on condition B, achieving similar results on condition B as on condition A before finetuning and similar results on condition A as on condition B before finetuning. These results seem to suggest that the modules did not generalise over both conditions by finetuning them, but
Instead forgot about (parts of) condition A in favour of performing well on condition B. This is a well-known problem in neural networks, termed catastrophic forgetting (French, 1999; Goodfellow et al., 2013). An exception to this is the \texttt{QUERY[CYLINDER]} module, which achieves similar accuracy (approximately 66\%) on both experimental conditions after finetuning.

Table 5.8: Overall question answering accuracy on both conditions A and B before and after finetuning (FT) on condition B.

<table>
<thead>
<tr>
<th>Model</th>
<th>Before FT</th>
<th></th>
<th>After FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEP (Johnson et al., 2017b)</td>
<td>96.6</td>
<td>73.7</td>
<td>76.1</td>
</tr>
<tr>
<td>TbD (Mascharka et al., 2018)</td>
<td>98.8</td>
<td>75.4</td>
<td>96.9</td>
</tr>
<tr>
<td>NS-VQA (Yi et al., 2018)</td>
<td>99.8</td>
<td>63.9</td>
<td>64.9</td>
</tr>
<tr>
<td>NSCL (Mao et al., 2019)</td>
<td>98.8</td>
<td>98.9</td>
<td>/</td>
</tr>
<tr>
<td>Hybrid Procedural Semantics</td>
<td>99.4</td>
<td>70.7</td>
<td>71.3</td>
</tr>
</tbody>
</table>

To further test the effects of finetuning, the finetuned modules are evaluated through the CLEVR benchmark task. These results are summarised in Table 5.8 and compared against other work where the same experiment was conducted. The visual question answering accuracy goes from 99.4\% to 71.3\% on condition A and from 70.7\% to 94.4\% on condition B. This corresponds with the results obtained in the individual evaluation of the modules, reported in Table 5.7. In sum, the modules do not seem to generalise well over both experimental conditions even after finetuning.

5.6 Conclusion

In this chapter, I have introduced a novel methodology for visual question answering that combines insights from modular neural networks approaches and neuro-symbolic approaches, namely \textit{hybrid procedural semantics}. In hybrid procedural semantics, task-specific and modular neural networks are integrated in procedural semantic representations through a number of sub-symbolic primitive operators. Interwoven with purely symbolic primitive operators, these hybrid procedural semantic representations can compute the answer to a question given the raw input image. Compared to the state of the art (Section 5.2), hybrid procedural semantics offers a number of key benefits. First, it elegantly and flexibly combines the strengths of symbolic and sub-symbolic techniques through procedural semantic representations. The sub-symbolic primitives are implemented through \textit{neural modules}, where each module performs a single, atomic task and thereby captures a specific concept. The specialisation of modules not only enhances their transparency, it also allows to more easily retrace the source of reasoning errors. Apart from this, the neural modules are (i) trained independently from each other, allowing the repertoire of modules...
to remain open-ended, (ii) highly modular such that they can be freely combined, (iii) compact in terms of layers and trainable parameters, thus more data-efficient, and (iv) designed to produce transparent and interpretable intermediate results (Section 5.3). Through the integration with computational construction grammar, specifically the CLEVR grammar (see Section 3.4), nearly the entire visual question answering process becomes transparent, explainable and human-interpretable. The only exception to this are the modules’ internal decision processes. The experimental results in Section 5.5 have shown that the neural modules individually achieve near-perfect levels of accuracy and that the results obtained by hybrid procedural semantics on the CLEVR benchmark task are competitive with state-of-the-art results, with the main novelty of hybrid procedural semantics being the beneficial properties just discussed. However, experimental results have also shown that the neural modules do not generalise well to similar, yet unseen, feature combinations even with additional finetuning.

Although hybrid procedural semantics offers high levels of accuracy and operates directly on raw image data, I argue that the discrimination-based concept learning approach of Chapter 4 has different, yet highly valuable properties. First, the grounded concepts offer complete transparency and human-interpretablity, both in terms of the concepts and the entire reasoning process during visual question answering. While the neural modules have been specifically designed to focus on atomic tasks and to produce interpretable results, their internal decision making processes remain hidden. What exactly the neural modules have learned can only be investigated through additional testing, such as the generalisation experiment in Section 5.5.3. In contrast, the concepts from Chapter 4 explicitly list their discriminative features and prototypical values. Second, the grounded concepts can be learned using much fewer data than the neural modules. Whereas the neural modules require several epochs of training, the agents using the grounded concepts achieve communicative success after merely ~1000 interactions. Third, the neural modules allow for internal inconsistencies as separate modules need to be designed for performing different operations on the same concept. For example, both $\text{FIND}[\text{CUBE}]$ and $\text{QUERY}[\text{CUBE}]$ modules are required since different neural architectures are necessary for either finding cubes in images or identifying whether some part of an image is a cube. These modules may contradict each other, e.g. when a cube found by the $\text{FIND}$ module is rejected by the $\text{QUERY}[\text{CUBE}]$ module. This does not occur when using the grounded concept representation from Chapter 4. There, exactly the same concept could be integrated in both the $\text{FILTER}$ primitive and the $\text{QUERY}$ primitive. Fourth, the grounded concepts effortlessly generalise over unseen combinations of attributes, whereas the neural modules require an additional finetuning procedure. Even then, the performance of these modules suffers as they insufficiently generalise over the experimental conditions of the CoGenT dataset. Finally, in Chapter 4, I have shown how additional concepts can be added to the agent’s repertoire on the fly. This is not possible with neural modules. Here, the set of modules (i.e. concepts) needs to be specified and trained in advance. However, in comparison to other modular neu-
5.6. CONCLUSION

Hybrid network approaches, the repertoire of neural modules in hybrid procedural semantics can be more easily expanded without the need to retrain any existing modules. In sum, the hybrid procedural semantics approach is best suited for a specific task that remains static over time whereas the discrimination-based concept learning approach of Chapter 4, due to the properties outlined above, is more suited for autonomous agents that face a dynamic environment and require an open-ended set of concepts that can be acquired and successfully used in communication after only a few interactions.

In both this and the previous chapter, I have introduced methodologies for learning and grounding concepts and for integrating these concepts in (hybrid) procedural semantic representations. I have demonstrated these methodologies through the task of visual question answering on the CLEVR dataset. To obtain the procedural semantic representations underlying the CLEVR questions, I made use of the CLEVR grammar introduced in Chapter 3. This is a hand-written grammar that offers complete coverage of the questions in the CLEVR dataset. In the following chapter, I will take up the challenge of learning this grammar.

5.6.1 Contributions

Hybrid procedural semantics constitutes a contribution of this dissertation (C4) that is directly applicable in a wide range of intelligent systems, such as conversational agents, intelligent tutoring systems, and human-robot interaction systems. This methodology fully exploits the pattern recognition capabilities of neural networks, combined with higher-level reasoning capabilities of symbolic approaches. The modular neural networks, in particular, are designed as to maximise the open-endedness, transparency and learning efficiency of the entire system, while their integration with symbolic reasoning processes further maximises the flexibility and adaptivity of the approach. Hybrid procedural semantics directly contributes to the primary objective of this dissertation (O1), as it paves the way for future intelligent systems with more explainable and coherent reasoning capabilities, grounded in the domain of the application. In terms of visual question answering (O2), the hybrid procedural semantics methodology provides a radically different approach that achieves results that are competitive with the state of the art and possesses the aforementioned desirable properties.
Chapter 6

Learning Morpho-Syntactic and Semantic Structures

6.1 Introduction ......................................................... 184
6.2 Background and Related Work ..................................... 186
  6.2.1 Usage-based Language Acquisition .............................. 186
  6.2.2 Computational Models ........................................... 191
6.3 Methodology ....................................................... 196
  6.3.1 The Elicitation Game ............................................ 196
  6.3.2 Tutor and Learner ............................................... 198
  6.3.3 Environment .................................................... 200
  6.3.4 Interaction Script .............................................. 200
  6.3.5 Intention Reading with IRL ................................. 204
  6.3.6 Pattern Finding with FCG ............................... 209
  6.3.7 Competition between Repair Strategies ..................... 219
  6.3.8 Alignment ................................................... 222
  6.3.9 Tutor Behaviour ............................................. 223
  6.3.10 Learning Dynamics ......................................... 224
6.4 Experiments ....................................................... 225
  6.4.1 Metrics ....................................................... 225
  6.4.2 Main Results ................................................ 226
  6.4.3 Probabilistic Tutor Strategy ................................. 233
6.5 Conclusion ......................................................... 235
  6.5.1 Contributions ................................................. 236
6.1 Introduction

This chapter presents the third and final experiment of this dissertation. The experiment investigates how an agent can simultaneously acquire the morpho-syntactic and semantic structures underlying linguistic utterances and capture them in the form of a computational construction grammar. This is achieved through a novel methodology where the agent is endowed with two learning mechanisms inspired from usage-based theories of language acquisition, namely intention reading and pattern finding (Tomasello, 2003, 2009b). Intention reading allows to identify and reconstruct the goals and communicative intentions of an interlocutor. Pattern finding, on the other hand, allows to detect differences and similarities in sensorimotor experiences and create abstractions over them. Theoretical and empirical evidence for both intention reading (Bruner, 1983; Sperber and Wilson, 1986; Meltzoff, 1995; Nelson, 1998) and pattern finding (Goldberg, 1995; Croft, 2000; Diessel, 2004; Goldberg, 2006) is abundant (Doumen et al., forthcoming). In this experiment, however, I argue that the interplay of these cognitive capacities is crucial for bootstrapping language. Importantly, the goal of this experiment is not to construct a realistic model of child language acquisition. However, as these mechanisms are the only ones known to successfully bootstrap language, namely in children, I rely on them to operationalise an intelligent, autonomous agents and learn a successful communication system. Furthermore, by relying on mechanisms from child language acquisition, the agent’s communication system will exhibit the same beneficial properties, such as robustness, flexibility, adaptivity and open-endedness.

The mechanistic model of intention reading and pattern finding presented in this chapter is operationalised through a language game (see Section 2.2.2) in a tutor-learner scenario, where the agents are situated in scenes from the CLEVR dataset (Section 3.3). The tutor has an established grammar, specifically the CLEVR grammar (Section 3.4), whereas the learner starts with an empty linguistic inventory. The agents play an elicitation game where the speaker has a particular concept in mind and asks a question to the listener about the objects in their shared environment in order to elicit that concept. The correct concept, i.e. the speaker’s intention, is provided as feedback in the end. This game effectively models an interactive version of the CLEVR visual question answering task. The learning problem involved is twofold. First, the meanings underlying the observed questions need to be reconstructed by relying solely on the provided answer and the observed scene (i.e. intention reading). Second, there is the generalisation over constructions based on pairs of observed questions and reconstructed meanings (i.e. pattern finding). The interplay of intention reading and pattern finding tackles these learning problems and allows the agent to gradually acquire a productive grammar, consisting of form-meaning mappings, that can be used for both language comprehension and production, without ever having observed the meanings.

Intention reading is used to reconstruct the meaning underlying the observed question,
given the provided intention and the scene. In other words, the agent engages in a process of meaning creation. This is implemented through Incremental Recruitment Language (IRL) (Section 2.4), specifically by the composition of semantic networks (Section 2.4.5) using the primitive cognitive operators designed for the CLEVR grammar (Section 3.4).

Pairing the reconstructed meaning with the observed question yields a form-meaning mapping, or a construction. Initially, the agent does not know which parts of the form correspond to which parts of the meaning. Hence, the form-meaning mapping is stored holistically. After observing more and more form-meaning mappings, pattern finding is used to abstract over re-occurring form-meaning patterns and to capture the compositional structure of language via a network of grammatical categories. Constructional language processing is implemented through Fluid Construction Grammar (FCG) (Section 2.3), with the categorial network (Section 2.3.5) playing a central role. The operationalisation of pattern finding strategies in FCG is an adaptation of the work by Doumen et al. (forthcoming).

The main challenge of this experiment lies in the vast search space of possible meanings intended by the tutor. Particularly, many different semantic networks can lead to any given concept in any given scene scene, but most of these networks do not capture the true meaning underlying the observed question and do not generalise to other scenes. I will show that the interplay of intention reading and pattern finding allows to overcome this challenge. Specifically, I show that intention reading facilitates pattern finding by providing meaning hypotheses and, in turn, pattern finding narrows down the search space faced by intention reading. Combined with mechanisms that model the entrenchment of constructions, this ensures that only constructions that can be used successfully in communication remain.

The mechanistic model of intention reading and pattern finding presented in this chapter constitutes the most important contribution of this dissertation. It not only provides computational evidence for the cognitive plausibility of these cognitive capacities, it also provides one of the first accounts on the computational representation, processing and learning of large-scale, bidirectional, and open-ended construction grammars through communicative interactions and, most importantly, chiefly contributes to the objectives of this dissertation by pushing forward the state of the art in the development of autonomous agents with human-like communication systems.

The remainder of this chapter is structured as follows. In Section 6.2, I first provide an overview of the field of usage-based language acquisition, which serves as the theoretical and empirical underpinnings of the experiment. Also in this section, computational models following the usage-based approach are discussed. I focus separately on models implementing pattern finding, intention reading or both capacities. Important differences and shortcomings are highlighted. The experiment itself is discussed in Section 6.3. This section gives a detailed description of the design and implementation of the experiment. It also explains in depth how intention reading and pattern finding are operationalised using
IRL and FCG, respectively. Afterwards, the evaluation metrics and experimental results are presented in Section 6.4. Finally, Section 6.5 highlights the main contributions of this chapter and discusses them in the broader context of this dissertation.

6.2 Background and Related Work

6.2.1 Usage-based Language Acquisition

The contributions of this dissertation are aligned with theories from cognitive science, developmental psychology and linguistics stating that language is a dynamic system in which all linguistic structures are emergent through a gradual process of communicative interactions using general cognitive processes (Peters, 1983; Hopper, 1987; Jasperson et al., 1994; Croft, 2001; Tomasello, 2003; Ambridge and Lieven, 2015) (see also Section 2.2.1). During this gradual process, linguistic structures become more abstract, organised and efficacious for serving the main purpose of language: to communicate. Notably, Tomasello (2003, 2009b) identifies two cognitive processes that play a crucial role in learning language: intention reading and pattern finding.

Intention Reading

Intention reading refers to the functional or semantic dimension of linguistic communication (Tomasello, 2003). It is a set of skills that allows to discover the communicative intentions of the interlocutor and thereby learn the linguistic conventions they use through the process of cultural transmission. This skill set includes, among others, understanding animate action, understanding the pursuit of goals and means to a goal, sharing attention to objects and events, actively directing attention through gestures, etc. Intention reading allows to learn the intentions of others, imitate them and predict them. Children continuously use these processes to make sense of the situation, taking into account their environment, the interlocutor and past experiences.

Intention reading is one aspect of what is called shared intentionality (Tomasello et al., 2005; Tomasello and Carpenter, 2007). Shared intentionality encompasses the ability to partake in collaborative activities having shared goals and intentions. It is identified as a crucial difference between human cognition and that of other species (Tomasello et al., 2005). Next to intention reading, shared intentionality also requires cultural learning skills, a motivation to share psychological states with others and cognitive representations that are sufficiently well-developed for doing so. In turn, shared intentionality enables cultural cognition and evolution, which then allows for the establishment of social constructions and institutions, e.g. language, maths, marriage, government, etc. The development of shared intentionality happens during a child’s first 14 months (Tomasello et al., 2005). Two pathways of development, which are very closely related, have been identified:
• **Pathway 1.** The ability to understand others as animate, goal-directed and intentional agents.

• **Pathway 2.** The motivation to share emotions, experiences and activities with others.

For both of these pathways, children go through several stages of development.

**Pathway 1.** The first pathway explains how children learn to understand intentional actions. Already a few months after birth, infants can tell the difference between an animate action, produced by some actor, and inanimate, caused motions (Bertenthal, 1996). Together with following the actor’s gaze (D’Entremont et al., 1997), this forms the basis of understanding intentional actions. By 10 months of age, children learn to recognise goals. They understand that an actor is pursuing a certain goal and persists until the goal is reached (Gergely et al., 1995; Behne et al., 2005). They also understand the emotional response that follows when a goal is or is not reached (Behne et al., 2005). Recognising goals also implies an understanding of the actor’s perception. Indeed, the child understands that the actor perceives the environment and that this helps to guide the actor’s actions and to determine if its goal is reached or not (Moll and Tomasello, 2004). Finally, around their first birthday, children not only understand the pursuit of goals, but also recognise plans towards goals. In particular, the child understands that the actor considers several actions plans (Gergely et al., 2002) and that the actor attends to certain objects in order to reach the goal (Tomasello and Haberl, 2003). Understanding intentional action allows for cultural, imitative learning. Children learn that when they have the same goal, they can use the same means as observed before to reach their goal. This is not only useful for predicting what others will do, but also learning how things are done conventionally.

**Pathway 2.** The second pathway explains how children learn to share (and want to share) intentions with others. Similar to the first pathway, three stages of development can be identified. Not coincidentally, these stages occur around the same age as those of the first pathway. After a few months, children share and exchange emotions with interlocutors through so-called protoconversations (Hobson, 2002). This not only requires the understanding that the interlocutor is an animate agent, but also the motivation and cognitive capacity for doing so. Later on, around 10 months, children share goals and perceptions. Their gaze becomes coordinated with the interlocutor and shared goals serve to coordinate actions, e.g. building a tower of blocks together (Hay, 1979; Hay and Murray, 1982; Verba, 1994). Finally, around 12 months, children not only understand the shared goals, but also the role of each participant. There is a motivation to help the other fulfil its role (Ross and Lollis, 1987; Warneken et al., 2006) and a deeper understanding of intentional actions allows for role-reversal (Carpenter et al., 2005). Also, perception becomes even more coordinated and the child actively attempts to establish joint attention, e.g. by pointing (Liszkowski et al., 2006).
CHAPTER 6. LEARNING STRUCTURES

Relation to Language. The development of shared intentionality culminates in children’s first linguistic communication. Many aspects of the aforementioned pathways can also be recognised in language. First, linguistic symbols can be seen as bidirectional coordination devices. Children understand and learn to play both roles: speaker and listener. Role reversal and imitation are crucial as children learn to use linguistic symbols towards others in the same way as others have used them. Similarly, coordinated perception and joint attention allows children to learn that people attend to particular things and can express this in various ways through language (Clark, 1997; Tomasello, 2009a). Second, communication is a collaborative activity (Pickering and Garrod, 2004). There is a joint goal to reorient the listener’s attention such that it aligns with that of the speaker. Both participants are aware of their own and each others’ role and actively collaborate in pursuit of the common goal. In particular, the speaker collaborates by expressing its intention in a way that is potentially comprehensible for the listener. In turn, the listener collaborates by following the gaze or pointing gestures of the speaker, by making inferences or by asking for clarification (Golinkoff, 1993). In all of this, the capacity to read and share intentions is at the foundation. Intention reading makes it possible to understand goals and means to reach them. Together with the desire to share these goals and intentions, this allows for the emergence of linguistic communication and many other skills of cultural cognition.

Pattern Finding

Pattern finding refers to the structural or grammatical dimension of linguistic communication (Tomasello, 2003). It allows to detect patterns in sensorimotor input and create abstractions over these patterns. This includes, among others, categorisation on both the perceptual and the conceptual level, schema-formation from recurrent patterns, the creation of analogies and statistical distributional analyses of perceptual sequences. From the perspective of construction grammar, the process of pattern finding allows to create compositional generalisations over constructions, taking both the form and the meaning into account. Tomasello (2003) identifies several stages of generalisations children go through when learning language.

Learning Holophrases. Children’s early linguistic inventory consists of holophrase constructions. These are idiom-like, holistic constructions mapping the entire form that was observed directly to the entire meaning that could be reconstructed through intention reading. Holophrase constructions can correspond to single words, such as “birdie!” or “hold!”, or multi-word expressions that are compositional in adult speech but not so in early child language, e.g. “there-ya-go” or “lemme-see” (examples from Tomasello (2006, 2009b)). Both the form and the reconstructed meaning are kept as a whole.

Generalising over Holophrases. In a later stage of linguistic development, children learn item-based constructions. These are compositional generalisations over holophrase
constructions. They emerge through the processes of pattern finding by looking for similarities and differences across holophrase constructions, considering both the form and the meaning. In particular, the child discovers that some part of the form of the holophrase corresponds to some part of its meaning. The differences across holophrases can be extracted and stored as separate constructions. The parts of form and meaning that are similar make up a pattern with one or multiple slots. These slots can be filled by other constructions, such as the differences that were just extracted, but also others. This mechanism is illustrated in Figure 6.1. In this example, the child already knows a holophrase construction mapping the form “dog wants ball” to its meaning. When the child observes the utterance “dog wants food” and reconstructs its meaning through intention reading, the pattern finding capacity becomes active. A minimal difference between these holophrase constructions is found and extracted, leading to a FOOD construction and a BALL construction. Additionally, a pattern of the form “dog wants ?X” is created, which captures some wanting event by the dog. It has a single slot, denoted by ‘?X’ on the form side, that can be filled with various items that act as the object in the wanting event. Slots in item-based constructions are not only intended for lexical constructions, like in this example. Similar processes of abstraction are used for further generalisations of item-based constructions. Continuing the example of Figure 6.1, further abstraction could lead to a ?X wants ?Y construction and ultimately to a fully abstract transitive construction ?NP-1 ?V ?NP-2. In general, further abstraction would allow slots to be filled by again using item-based constructions, e.g. to construct the form “the lazy brown dog” for ‘?NP-1’. Hence, constructions of varying degrees of abstraction can be used to fill slots.

Grammatical Categories. Not all constructions are equally likely to fill a particular slot of an item-based construction. While the slot of the item-based construction DOG WANTS ?X might be tied to BALL and FOOD, it is highly unlikely that the RED construction will be used instead. The distribution of item-based slots and their fillers can be represented in the form a network. Such a network essentially captures the grammatical categories underlying an individuals grammatical knowledge. Just like constructions, grammatical categories are emergent and are acquired in a gradual process through communicative interactions and pattern finding (Pine and Lieven, 1997; Croft, 2001). In the example above, grammatical categories emerge both for the slot of the item-based construction DOG-WANTS ?X and for the lexical fillers BALL and FOOD. Both lexical fillers are linked to the slot in the network of grammatical categories.

Intention Reading and Pattern Finding

The interplay of these cognitive processes in the context of usage-based language acquisition is crucial. Intention reading, defined as reconstructing the interlocutor’s intended meaning, is extremely difficult. Indeed, the possibilities faced by intention reading consists of every possible meaning that could be intended by the interlocutor. To make this
CHAPTER 6. LEARNING STRUCTURES

Pattern Finding

"dog-wants-food"

Intention Reading

Khown construction

"dog-wants-ball"

Known construction

"dog-wants ?X"

"food"

Figure 6.1: Considering both form and meaning, pattern finding looks for differences and similarities across holophrase constructions. The differences are extracted and stored as separate constructions: the FOOD construction and the BALL construction. The remainder constitutes a usage pattern with one or multiple slots: the DOG-WANTS ?X construction.

Partial Analysis

“cat-wants food”

"cat-wants ?X"

“food”

Figure 6.2: Intention reading and pattern finding are highly complementary. The FOOD construction, created earlier through pattern finding, provides constraints on the intention reading process when observing the unknown utterance “cat wants food”. In turn, pattern finding uses the utterance and the reconstructed meaning to create a new pattern.
feasible, intention reading relies on many different sources of information: the observed utterance, the current environment, past experiences, the linguistic inventory, the interlocutor’s gaze, actions and emotions, shared cultural background, etc. As intention reading is about reconstructing meaning, it also facilitates the creation of constructions. Indeed, a construction requires both a form and a meaning. While the form is typically observed, the interlocutor’s intended meaning has to be reconstructed through intention reading. A crucial source of information for intention reading is provided through pattern finding. Indeed, the process of pattern finding allows to create abstractions over constructions, taking both the form and the meaning into account. These abstractions can already provide a partial understanding of the utterance or, in other words, a partial reconstruction of the intended meaning. This provides constraints on the intention reading process, as it should take this partial meaning into account. After completing the partial meaning, pattern finding can again be used to create other constructions and abstractions over them. This mechanisms is illustrated in Figure 6.2. Observing the utterance “cat wants food”, the previously acquired FOOD construction provides a partial understanding of the utterance. In other words, some of the meaning of the utterance is already known. In order to reconstruct the intended meaning of the utterance, the process of intention reading make use of this additional information. Specifically, the partial meaning provides constraints on the enormous search space of possible meanings. Once intention reading is finished, pattern finding can again take over and create a pattern in which the FOOD construction can be used. Depending on the state of the linguistic inventory, various patterns are possible, as illustrated on the right side of the figure.

6.2.2 Computational Models

In prior work, three groups of computational models can be identified: models focusing exclusively on intention reading, models focusing exclusively on pattern finding and models that incorporate both cognitive capacities. In what follows, I discuss each of these groups in more detail.

Intention Reading

Computational models of intention reading are mostly situated in the field of robotics. Various studies on intention reading make use of a robotic arm that is situated on some playing field and equipped with various sensors and actuators. The sensors are used for observing the playing field and the interlocutor, being either another robot or a human, while the actuators serve to control the robot arm and make changes in the playing field. Various cognitive processes that can be attributed to intention reading, as discussed in Section 6.2.1, are studied using this methodology.

In the work by Vinanzi et al. (2019, 2020), the robotic arm is trained to recognise intentions. Specifically, the robot’s task is to learn from task demonstrations and subsequently
identify the correct intention as fast as possible. To achieve this, the robot observes a human
interlocutor arrange four coloured blocks in a particular order. After learning from
various such observations, the robot should be able discover the ordering of the blocks
intended by the human demonstrator as soon as possible. This is learned in a three step
process. First, skeleton data points and eye gaze direction are represented in a common
feature space. Next, (hierarchical) clusters are identified in that feature space, representing
key combinations of postures and eye gaze directions. Finally, transitions between
these clusters are learned. A particular sequence of clusters will then be labelled as a par-
ticular intention. At test time, the model will then estimate probabilities for each of the
possible intentions (i.e. arrangement of blocks), given the already observed postures and
eye gaze directions. The model is evaluated on both its accuracy and the time required to
make a prediction.

A similar experimental setup is used by Jansen (2006) and Jansen and Belpaeme (2006a,b)
to study the imitation of intentions. Here, two robotic arms interact with each other where
one is a tutor and the other is a learner. An intention is expressed as a set of predicates
concerning the arrangement of three blocks (A, B and C) on a two-dimensional grid, e.g.
\(\text{above}(C, A) \land \text{left-of}(A, B)\). The learner should not only correctly identify the tutor’s in-
tention, but also correctly imitate it. In line with findings from the psychological literature,
the imitation behaviour focuses on the goal and not on the exact trajectory towards that
goal (Bekkering et al., 2000; Gleissner et al., 2000; Wohlschläger et al., 2003). In other words,
the actions performed by the robot can be completely different, as long as the goal state is
the same. Even more, the robots are given two different, yet equally expressive, represen-
tations for expressing goals (e.g. one uses \(\text{above}\) while the other uses \(\text{below}\)) and two
distinct playing boards with different starting configurations. This avoids that the learner
merely copies the tutor’s end state or the tutor’s actions and guides the model towards
learning the tutor’s true intentions. Jansen (2006) and Jansen and Belpaeme (2006a,b) em-
ploy an interactive task learning methodology, learning new representations only when
necessary and rewarding representations that are used successfully while punishing their
competitors. They show that the learner can successfully imitate the tutor under various
experimental conditions.

Finally, Dominey and Warneken (2011) incorporate even more aspects of intention reading
in their computational model, again using a robotic arm platform. Not only is the robot ca-
pable of recognising and imitating intentions, it can also actively cooperate in a given task
or perform a task with role reversal. These skills are also attributed to intention reading
capabilities as shown by Warneken et al. (2006) for cooperation and Carpenter et al. (2005)
for role reversal. Concretely, the robot incorporates these skills by storing intentions as a
series of actions and representing each action in a so-called “we intention” structure. This
structure can be easily transformed into a “me intention” for executing the action, or a “you
intention” for recognising the action. Cooperation amounts to first matching the observed
actions to an intention and then predicting and executing the following action(s). Role
reversal can be implemented by ‘replaying’ the same action sequence with each intention transformed into the opposite perspective (i.e. “me intention” becomes “you intention” and vice versa). Dominey and Warneken (2011) report five successful runs of each of the following experimental conditions: recognition, imitation, cooperation and role reversal.

While the models outlined above represent crucial steps in understanding the various aspects of intention reading, none of them study intention reading capabilities in relation to language. Even though the model by Dominey and Warneken (2011) includes a linguistic component, this is merely used to steer the control flow of the task. The linguistic structures used in this case are given a priori. None of them are learned. This is where pattern finding could be used.

**Pattern Finding**

In this section, I discuss computational models for the pattern finding capability. Specifically, I only consider models that learn form-meaning mappings. Models focussing only on form-patterns are less relevant for the purposes of this chapter. Here, form-meaning mappings are conceived in the broadest possible sense, ranging from phonemes and words on the form side, to procedural semantics, first order logic or distributional semantics on the meaning side. I discuss three groups of pattern finding models as identified by Doumen et al. (forthcoming).

One approach to model pattern finding is by extracting constructions from annotated corpora through inductive learning methods. Such methods have been applied using either parse trees (Zuidema, 2006) or utterances annotated with POS tags, semantic tags and dependency relations (Dunn, 2017, 2018) as their input. These inductive learning methods capture a minimal grammar that has maximum coverage over the corpus. However, the resulting grammars cannot be used for language comprehension and production, making these models irrelevant for an autonomous agent, as is considered in this chapter.

The model by Gaspers et al. (2011) studies construction learning under referential uncertainty. Specifically, the model makes use of the RoboCup Soccer corpus. This corpus provides utterances accompanied by a description of the situational context, represented as a set of logic predicates. One of the predicates corresponds to the meaning of the utterance. For example, the utterance “purple10 kicks to purple7” is accompanied by \{ballstopped, badPass(pink1, purple10), ..., pass(purple10, purple7)\}. The task of the agent is twofold: to learn which of the predicates corresponds to the utterance and to learn item-based constructions over patterns occurring in the utterances and the predicates. Next to constructions, the model also learns a network of associations between lexical items and slots in item-based constructions, corresponding to a system of grammatical categories. From the example above and other observations, the item-based construction mapping “E1 kicks to E2” to pass(E1, E2) could be learned. These mappings are learned through probabilistic cross-situational learning. Gaspers et al. (2011) initially demonstrate their methodology
starting from the word level. In later versions, the starting point is changed to graphemes and phonemes (Gaspers and Cimiano, 2012, 2014; Gaspers et al., 2016). The problem definition specified in these models, i.e. learning constructions under referential uncertainty, falls outside the scope of this chapter. Specifically because it is assumed that the intention reading process will reconstruct the meaning of an observed utterance without such uncertainty. Additionally, the utterances used in the corpus are rather short and always correspond to a single predicate of the situational context, limiting the applicability and scope of the approach. In this chapter, I go beyond single predicate meanings.

The models that are most relevant for this chapter are those learning from a corpus of utterances annotated with their meaning representation and resulting in productive grammars. Gerasymova and Spranger (2010, 2012) and Gerasymova et al. (2012) use the language game methodology and FCG to implement a single agent capable of learning holophrases, item-based constructions and abstract constructions for the aspectual marking system in Russian. The agent in their model observes Russian utterances paired with a semantic annotation of their temporal event structure. The same methodology is applied by Beuls et al. (2010) to the conjugation of verbs in Hungarian, specifically on the agreement marking system. Extending this approach to a population of agents, Van Eecke (2018, Ch. 7) studies the emergence of a range of constructions of varying degrees of abstractions along with a network of grammatical categories. In these experiments, the population of agents has to agree on which word order to use for primitive noun phrases in English. The experiments rely on FCG’s meta-level architecture (Van Eecke and Beuls, 2017) and generalisation and specialisation operators that create new constructions based on existing constructions and observations (Van Eecke and Beuls, 2018; Van Eecke, 2018). Using these, the population of agents converges to a stable solution. Other such learning operators have been introduced by Chang (2008). Starting from an initial inventory of lexical constructions, novel constructions are learnt either from input data, associating an observed form with its meaning, or by reasoning over existing constructions. Three such operators for recombining structural elements of existing constructions are defined: 1. merging (‘throw-block’ + ‘throw-ball’ → ‘throw-toy’), 2. joining (‘human-throw’ + ‘throw-bottle’ → ‘human-throw-bottle’) and 3. splitting (‘throw-frisbee’ + ‘throw’ → ‘frisbee’). Finally, Dominey (2005a,b, 2006) makes use of neural network techniques for the acquisition of holophrase, item-based and abstract constructions centred around argument structure relations. Starting with the ability to differentiate closed-class and open-class words, the model learns a mapping between slots in the argument structure construction and their semantic roles. While all of these models have studied interesting ideas, either they are focused on a very specific linguistic phenomenon, or the complexity of the input data is rather limited. Also, all aforementioned models have access to additional information next to the raw utterances and the meaning representations, e.g. a precomputed segmentation of the utterance, a predefined lexicon or a collection of predefined grammatical categories.
6.2. BACKGROUND AND RELATED WORK

Intention Reading and Pattern Finding

In the third group of computational models, both intention reading and pattern finding are studied. Spranger and Steels (2015), Spranger (2015) and Spranger (2017) apply the language game methodology to a single agent learning English spatial expressions from a tutor agent. The tutor actively guides the agent’s learning process by gradually providing more difficult examples in more difficult scenes. This could be compared to child directed speech. On the one hand, the techniques introduced by Gerasymova and Spranger (2012) for learning holophrase constructions, item-based constructions and abstract constructions with FCG are extended, allowing more fine-grained semantic-based generalisations. However, a domain-specific hierarchy of semantic categories on which the generalisations are based is given in advance. For example, the meanings of lexical items such as ‘near’, ‘far’, ‘front’ and ‘back’, ‘north’ and ‘south’ etc. are categorised as proximal, projective and absolute spatial relations respectively. Additional subcategories such as horizontal and vertical projective relations are given as well. On the other hand, they introduce mechanisms for reconstructing the meaning of an utterance based on the interlocutors intention. This reconstruction process uses IRL and consists of a heuristic search process. During the search process, a network of IRL predicates is expanded incrementally until it leads to the interlocutors intention. While these models are most relevant for the purposes of this chapter, similar to the pattern finding models, the linguistic domain on which they are applied is rather limited. For example, the tutor agent uses a repertoire of only seven meanings to express the various spatial expressions (Spranger and Steels, 2015). Additionally, various scaffolds are put in place to aid learning, such as the provided hierarchy of spatial categories. Finally, comparison with or reproduction of these models was not feasible because of insufficient methodological details.

In general, the methodology presented in this chapter aims to push forward the state of art as described above in three ways. First, the methodology presented here operationalises both intention reading and pattern finding, and importantly, also the cooperation between these processes. Second, the experiment operates on a much larger scale compared to the aforementioned models and does not focus on a specific linguistic phenomenon, such as the Russian aspectual system (Gerasymova and Spranger, 2010), the Hungarian agreement system (Beuls et al., 2010) or English spatial language (Spranger and Steels, 2015). The CLEVR dataset, as used in this experiment, is much larger in comparison to those used in previous work and contains utterances of considerable complexity, both in terms of morpho-syntax and semantics. Third, the methodology is such that there are as few scaffolds as possible. The agent does not receive a predefined lexicon as in Beuls et al. (2010) or a taxonomy of semantic categories that guides the generalisation process of constructions as in Spranger and Steels (2015). While the agent does receive a predefined repertoire of semantic concepts, these are not used during the pattern finding processes.

¹The methodologies presented in Chapters 4 and 5 show how to learn such a repertoire of concepts.
Apart from that, the agent only receives a number of primitive cognitive operators. All other structures used by the agent, being constructions, semantic networks and a network of grammatical categories, can be acquired from scratch using general learning operators.

6.3 Methodology

This section describes all aspects of the language game experiment in depth. In Section 6.3.1, I first provide a high-level overview of the elicitation game and sketch the main difficulties of the learning problem faced by the agent. Subsequent sections dive deeper into the design and implementation of the game. This includes the design of the tutor and the learner agent (Section 6.3.2), the environment in which they operate (Section 6.3.3) and the interaction script they follow (Section 6.3.4). Afterwards, I discuss the three learning mechanisms taking place on the agent’s meta-level. I start with the implementation of intention reading (Section 6.3.5), which is an extension of IRL’s mechanism for goal-oriented composition of procedural semantic representations (see Section 2.4.5). Next, I discuss the implementation of pattern finding with FCG (Section 6.3.6), which is based earlier work by Doumen et al. (forthcoming). The pattern finding strategies of Doumen et al. (forthcoming) are adapted to incorporate intention reading. Section 6.3.7 discusses how these learning mechanisms are organised in the agent’s meta-level. The third learning mechanism, namely alignment, for modelling the entrenchment of constructions is discussed in Section 6.3.8. Section 6.3.9 discusses the strategies used by the tutor. Finally, with both agents, their environment, the interaction script, learning mechanisms and tutoring strategies in place, I provide an overview of the learning dynamics in Section 6.3.10.

6.3.1 The Elicitation Game

The goal of the elicitation game is to allow an agent to learn the morpho-syntactic and semantic structures underlying linguistic utterances in the form of a construction grammar and use this grammar successfully in communication. This is achieved by having two agents, a tutor and a learner, partake in a series of scripted, communicative interactions modelling the task of visual question answering. The learner is endowed with three learning mechanisms, namely the cognitive capacities of intention reading and pattern finding together with entrenchment dynamics.

High-Level Overview

The elicitation game goes as follows. The speaker has a particular concept in mind and wants to elicit this concept as a reaction from the listener by asking it a question about the scene. The listener’s task is to try and understand the question and provide the answer. Both tutor and learner can take on either of the discourse roles in the game. At the end of the game, regardless of their discourse roles, the tutor reveals the true answer to the
question. The success of the interaction is determined by checking whether the listener’s answer is the same as the concept that the speaker had in mind.

To make this game more concrete, imagine a parent and a child browsing an animated children’s book. On a page with drawings of animals, the parent would ask the child “What sound does the cow make?”. The parent is not trying to learn about animal sounds, but instead trying to elicit a particular reaction from the child. From the child’s perspective, however, the parent simply wants an answer to the question it just asked. If the child replies by shouting “moo!”, the parent gives a positive reaction. However, if the child doesn’t know or replies with “woof!”, the parent replies by providing the correct answer. The parent has revealed its intention. In both cases, the child realises that the parent actually wanted the child to reply “moo!”, which allows the child to learn something about the utterance “What sound does the cow make?” and its underlying meaning based on the current setting and the provided answer “moo!”.

In the elicitation game, the learner uses intention reading to reconstruct a possible meaning that could underlie the observed utterance, such that it leads to the provided answer in the current scene. In the example, the child mentally constructs a constellation of cognitive operations that leads to “moo!” in the current environment. For example, first focus on the book, then find the animal with black and white hair and then think about the sound it makes. Pairing the reconstructed meaning with the observed utterance constitutes a form-meaning mapping or a construction. Pattern finding is then used to generalise over this and other form-meaning mappings, e.g. “What sound does the dog make?”, and its reconstructed meaning, resulting in constructions of varying degrees of abstraction. Specifically, over the course of multiple elicitation games, the learner transitions from holophrase constructions to compositional item-based and lexical constructions, while a network of grammatical categories that captures the compositional relations between those constructions emerges. At the same time, the entrenchment dynamics ensure that only form-meaning mappings leading to successful communication remain.

When the learner acts as the speaker in the elicitation game, this serves as a kind of hypothesis testing. The concept revealed by the tutor in the end then allows the learner to validate or reject its hypothesis, namely the constructions that it used in producing its question. This is also taken care of by the entrenchment dynamics.

Main Challenges

There are two main challenges in the elicitation game, namely (i) to constrain the search space of intention reading and (ii) to overcome the acquisition of incorrect form-meaning mappings through pattern finding. Intention reading constitutes an enormous search problem since it faces the search space of all possible meanings that could be intended by the tutor. This space is vast and difficult to navigate. Therefore, the learner has to use as many resources as possible, such as the observed utterance, the answer (or more generally, the
intention) revealed by tutor, the current environment, past experiences and known constructions. A crucial aspect in narrowing down this search space even further is the interplay between intention reading and pattern finding. Specifically, acquired item-based or lexical constructions can provide a partial understanding of a previously unobserved utterance. This results in a partial meaning that provides constraints on the intention reading process. Indeed, part of the meaning is already provided and intention reading should now reconstruct the remainder. Crucially, it is not guaranteed that the reconstructed meaning will be correct, e.g. as it may focus on particularities of the current scene and thus fail to generalise to other scenes. Consequently, incorrect form-meaning mappings may still be learned. The mechanisms modelling the entrenchment of constructions overcome this by ensuring that over the course of many elicitation games only constructions that are used successfully in communication remain and unsuccessful constructions gradually disappear.

6.3.2 Tutor and Learner

The elicitation game requires the modelling of a tutor agent with an established grammar, and a learner agent that starts with an empty linguistic inventory.

Tutor

The tutor’s linguistic inventory is the CLEVR grammar (Section 3.4). This grammar not only offers complete coverage of all questions from the CLEVR dataset, it’s procedural semantics also allows the tutor to correctly compute all answers. Importantly, the tutor uses a subset of this grammar for the elicitation game. Specifically, all questions which have a meaning representation using cognitive operators that concern comparison, spatial relationships and logical operations have been left out. The main reason for this is that these are more complex cognitive operators that are often associated with longer and more complex utterances. As this experiment models processes involved in child language acquisition, it is more logical to start with the simplest utterances, as an adult would do when speaking to a child. For the CLEVR dataset, the simplest questions are on attribute identification (e.g. ’What color is the large metal cube?’), counting (e.g. ’How many tiny red spheres are there?’) and existence (e.g. ’Is there a small rubber cylinder?’). Table 6.1 shows an overview of the primitive cognitive operators necessary for these types of questions. The complete overview of cognitive operators in the CLEVR grammar can be found in Section 3.4.2.

Learner

Next to the subset of primitive operators shown in Table 6.1, the learner is given three learning mechanisms and a collection of semantic concepts from the CLEVR dataset. This collection includes the various colours, shapes, sizes and materials present in the CLEVR
Table 6.1: Subset of the primitive cognitive operators from the CLEVR grammar used in the elicitation game.

<table>
<thead>
<tr>
<th>Primitive Cognitive Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>(get-context ?context)</td>
</tr>
<tr>
<td>(query ?value ?object ?attribute)</td>
</tr>
<tr>
<td>(count ?number ?set)</td>
</tr>
<tr>
<td>(exist ?boolean ?set)</td>
</tr>
<tr>
<td>(unique ?object ?set)</td>
</tr>
</tbody>
</table>

dataset in a symbolic representation, but also numbers and booleans. These concepts are necessary for evaluating the semantic networks as they serve as the semantic entities used in IRL (see Section 2.4.3). These concepts are given as a scaffold because the main focus of this experiment lies on the acquisition of morpho-syntactic and semantic structures. However, they could also be learned using the methodology outlined in Chapter 4. Alternatively, the primitive operators could be operationalised through hybrid procedural semantics, thereby also capturing the semantic concepts, as outlined in Chapter 5.

All of the learner’s linguistic knowledge is captured in its construction inventory, which is initially empty. The learner only considers holophrase constructions, lexical constructions and item-based constructions. Constructions keep a score between 0 and 1, reflecting their entrenchment. These scores are based on past communicative success and affect the order in which constructions are considered during language processing. Specifically, constructions with a higher score are preferred over constructions with a lower score as they were used more successfully in the past. By increasing or decreasing these scores, the learner can shape its construction inventory. Put together, this constitutes a positive feedback loop between the success and use of constructions. The slot-and-filler relations between item-based constructions and lexical constructions are captured in a network of grammatical categories (see Section 2.3.5). Similar to constructions, links in the categorial network keep a score, reflecting the association strength between a slot and its filler. However, these scores are based on frequency rather than success. Specifically, the score of a link is increased whenever that link is used in a successful comprehension or production process and it is never decreased. These scores are not considered during language processing. Nodes in the categorial network with a similar distribution of links can be seen as similar grammatical categories. The similarity between categories can be quantified using weighted cosine similarity. Modelling the entrenchment of both constructions and grammatical categories corroborates findings from psychological and linguistic research showing that this plays an important role in the formation of one’s linguistic inventory (Doumen et al., forthcoming), see e.g. Langacker (1987); Schmid (2007); De Smet (2017); Theakston (2017).
6.3.3 Environment

The agents are situated in the scenes from the CLEVR dataset. A full description of these scenes can be found in Section 3.3. In the elicitation game, the 15,000 scenes from CLEVR’s validation split are used. The agents obtain symbolic representations of the scenes. In other words, they can perfectly categorise the size, colour, material and shape of each object in terms of the semantic concepts given to them. An example scene together with the agents’ internal representation of it is provided in Figure 6.3.

![Scene Example]

Figure 6.3: An example scene from the CLEVR dataset containing four objects (left) and the agents’ internal representation of it (right).

6.3.4 Interaction Script

Following the semiotic cycle (see Section 2.2.2), both the speaker and the listener go through a number of processes during every interaction of the elicitation game. Figure 6.4 provides a schematic visualisation of the interaction script, with all of processes numbered from 1 to 8. In the following paragraphs, each process of the interaction script is discussed.

Step 1: Role Selection (both agents)

The discourse roles of speaker and listener are randomly assigned to the tutor and the learner.

Step 2: Scene Selection (both agents)

A random scene from the validation split of the CLEVR dataset is selected and both agents receive a symbolic representation of it (see Figure 6.3).

Step 3: Topic Selection (speaker)

The speaker randomly selects the topic for the interaction. The topic can be any semantic concept from the CLEVR dataset. Concretely, this can be an attribute of an object (colour, size, material or shape), a number between 0 and 10, ‘yes’ or ‘no’. The only constraint on topic selection is that there should exist a question about the current scene having the topic as the answer.
Figure 6.4: A schematic visualisation of the interaction script of the elicitation game.
**Step 4: Conceptualisation (speaker)**

After having selected a topic, it is now the task of the speaker to come up with a question that has the topic as the answer. This process, called conceptualisation, is handled differently by the tutor and the learner.

For the learner, conceptualisation is implemented using IRL’s composer mechanism (see Section 2.4.5). The composer constructs a meaning network that satisfies a particular communicative goal. In this case, the goal of the speaker is to lead the listener to the topic in the current scene. Thus, it constructs a semantic network that, when executed, results in the topic.

The tutor, on the other hand, takes a short-cut when it comes to both conceptualisation and formulation (the next step in the interaction script). Specifically, the tutor randomly samples a question from the validation split of the CLEVR dataset that has the selected topic as the answer. This information is computed prior to the experiment. Taking into account the available primitives (Table 6.1), this results in 10,044 unique questions that can be uttered by the tutor.

This short-cut results in certain artefacts that stem from the design of the CLEVR dataset. For instance, the tutor will not always utter the most optimal question leading to a particular topic. Questions can contain superfluous words to identify some referent, e.g. asking about “the small blue cube” when there are only small objects in the scene and hence “the blue cube” would have been sufficient. Similarly, the tutor does not take into account the most salient features or the most discriminative object to come up with a question. Both of these artefacts do not pose an issue for the elicitation game, but could lead to faster learning as the tutor would be more actively dampening referential ambiguities for the learner.

**Step 5: Formulation (speaker)**

The speaker uses its construction inventory to produce an utterance that expresses the meaning network constructed in conceptualisation. The process of formulation succeeds when no more construction can apply and the entire meaning network is consumed. The tutor does not need to run the formulation process since it uses the short-cut described above. For the learner, on the other hand, formulation may fail when its construction inventory is inadequate for expressing the conceptualised meaning. When this occurs, a diagnostic signal’s the problem and triggers a jump to the meta-layer (see Section 2.2.5). Here, the learning mechanisms described in later sections will become active and try to repair the problem. When successful, routine processing continues and the formulated utterance is passed from the speaker to the listener. Otherwise, the learner restarts the conceptualisation process, but further explores the space of possible meaning networks that lead to the topic in the current scene. The learner attempts to express each new con-
ceptualisation until one of them succeeds. When none of the conceptualisations can be expressed by the learner’s current construction inventory, the game ends and counts as a failure. There is no learning opportunity in this case, as the tutor cannot provide any useful feedback without an utterance.

**Step 6: Comprehension (listener)**

In comprehension, the listener uses its construction inventory to analyse the observed utterance. It is successful when no more constructions can apply and the entire utterance was consumed. When successful, comprehension results in a meaning network. Similar to formulation, comprehension may fail for the learner. Specifically, the utterance may be completely unknown or parts of it are known resulting in a partial analysis. In both of these cases, the learner signals failure to the tutor, and the agents proceed to feedback and learning (step 8).

**Step 7: Interpretation (listener)**

When comprehension was successful, the listener interprets the resulting meaning network with respect to its own observation of the current scene. This results in an action that is relevant for the communicative task. In this case, the listener computes the answer to the observed question.

**Step 8: Feedback & Learning through Intention Reading, Pattern Finding and Alignment (both agents)**

At the end of the game, the agents determine if their interaction was successful. This is the case when the listener’s answer corresponds to the topic (i.e. the semantic concept) that the speaker had in mind. Conversely, the interaction may fail in any of the following cases: (i) the speaker’s conceptualisation and production processes failed, (ii) the listener could not comprehend the utterance, (iii) the listener’s interpretation led to an incorrect concept or (iv) the listener’s interpretation failed to compute an answer all together. The outcome of the interaction provides insights to the learner about the constructions it used during the interaction, if any. The alignment mechanism will alter the scores of the constructions on this basis.

Regardless of the outcome of the game, the tutor provides feedback to the learner. This feedback is the correct answer to the uttered question, regardless of who uttered it. Concretely, when the tutor was the speaker, it provides the intended topic (step 3). Conversely, when the listener was the speaker, the tutor provides the answer to the question formulated by the listener, if any. Together with the observed utterance, the current scene, past experiences and known constructions, the tutor’s feedback allows the agent to learn after a failed interaction. Learning consists of acquiring new construction(s) and/or adding new links in the categorial network in order to remedy an incorrect construction that was used
during the interaction or allow to comprehend/produce the utterance in the first place. This is where the meta-level diagnostics and repairs operationalising intention reading and pattern finding come into play. Specifically, intention reading allows to hypothesise about a possible meaning underlying the observed question, relying on the current scene, past experiences and the tutor’s feedback. The reconstructed meaning allows for the creation of one or several constructions by pairing it with the observed utterance and applying pattern finding to this and previously acquired constructions. In what follows, the implementation of these various learning mechanisms will be discussed: intention reading in Section 6.3.5, pattern finding in Section 6.3.6 and alignment mechanisms in Section 6.3.8. Finally, in Section 6.3.10, I discuss how these three learning mechanisms work together and allow the learner to bootstrap a successful communication system in the form of a construction grammar.

### 6.3.5 Intention Reading with IRL

The goal of intention reading is to reconstruct a possible meaning that could underlie the observed question, taking into account the tutor’s intention, the current scene, past experiences and known constructions. It is implemented using a modified version of IRL’s composer mechanism (see Section 2.4.5). In Section 2.4, I argued that semantic networks in IRL are equivalent to constraint programs and that evaluating semantic networks corresponds to constraint satisfaction with respect to the agent’s world model and repertoire of semantic concepts. In a nutshell, the composer performs a heuristically guided search process that starts from an empty meaning network, recursively adds cognitive operators and links them together until the composed network satisfies a particular communicative goal. In the case of intention reading, the communicative goal for the listener is to reconstruct the tutor’s intention. In IRL-specific terms, it is to construct a constraint program in which there exists a data-flow that allows to infer the semantic concept provided by the tutor from the listener’s world model, concept repertoire and a memory of past experiences. The addition of a memory structure is crucial in operationalising intention reading, as opposed to the conceptualisation process which also relies on IRL’s composer mechanism.

#### Main Challenges

The main challenges of the elicitation game, outlined in Section 6.3.1, can be reformulated in terms of IRL and the composer mechanism. The first challenge is to manage the huge search space faced by intention reading, which is the space of all possible meanings intended by the tutor. In terms of the composer, the primitive operators, which are the building blocks of constraint networks, can combine in many different ways with the only restriction being the type definitions of their arguments. Even more, each intermediate constraint network is considered for constraint satisfaction which can return multiple solutions, depending on the bindings from the agent’s world model and concept repertoire
that can be made. Both the construction of constraint networks and constraint satisfaction are combinatorial processes in IRL.

The search space faced by the composer in this experiment can be analysed according to the description in Section 2.4.5. Concretely, the primitive inventory has size $n = 6$, with an average arity of $a = 2.16$, and the maximum program size $k = 7$ (see Table 6.1). The composer’s search space thus consists of $\sum_{i=1}^{k} \binom{n}{i} 2^{8(i,a)} = 1.3494104e30$ intermediate constraint programs that can be constructed. Crucially, this analysis allows to combine any primitive with any other primitive and does not take into account the type definitions of the arguments, which already drastically reduces the potential number of constraint programs.

The second challenge consists of intelligently revising incorrect constructions acquired via pattern finding. If it turns out, due to a failed interaction, that some construction is incorrect, the learner will want to create a new, competing construction that has the same form, but a different meaning. Intention reading is required to obtain a new meaning hypothesis. However, with its default configurations, nothing stops the composer from constructing exactly the same constraint network as the one that was just found out to be incorrect.

In what follows, I discuss the mechanisms that are put in place in the composer to overcome these two challenges. Specifically, node tests are used for the first challenge, a memory mechanism is introduced to overcome the second challenge and partial analyses help in tackling both challenges. The composer is a highly configurable system and not all aspects of it needed to be modified for implementing intention reading. Therefore, I also refer to Section 2.4.5, which offers a complete and in-depth description of the composer, including all configurations and their default settings.

**Node Tests**

Node tests are used to accept or reject intermediate constraint networks during the composition process before constraint satisfaction takes place. These tests are used to prune the search space of the composer. To operationalise intention reading, two node tests are put in place. A first node test restricts the maximum length of the composed constraint network. Any network that contains more than 10 constraints is rejected. This covers all possible constraint networks that are required for the experiment and avoids an exhaustive search of the massive space of constraint networks. A second node test is responsible for detecting duplicates in the composed constraint networks. If a certain constraint network was already constructed and did not satisfy the communicative goal, the same constraint network constructed via a different path in the search space will also not work.
Memory

To ensure that the composer does not reconstruct the same constraint network for the same utterance multiple times, the learner stores past experiences in a memory. Specifically, the learner remembers the observed question, its answer and the scene in which the interaction took place. These experiences are then consulted during intention reading. Specifically, the reconstructed constraint network should not only lead to the answer in the current scene, but it should also lead to the previously remembered answers in the previously remembered scenes where the same question occurred. The composer keeps expanding the network until this additional goal test is satisfied. Effectively, the learner’s memory allows it to perform mental simulation of similar past experiences to guide its intention reading process.

Figure 6.5: On first occurrence of the utterance (top), the learner composes the shortest constraint network leading to the answer (top right). This program is not applicable in a later scene (bottom left). To revise the program, the learner keeps track of past experiences, storing observed question, scenes and answers. A reconstructed constraint network is considered a solution when it can be satisfied in the current scene, but also in previous scenes where the same question was observed. This is true for the second constructed constraint network, shown on the bottom right.

The memory mechanism is illustrated in Figure 6.5. In the first scene, shown in the top left of the figure, the learner observes the question “What shape is the large purple metal object?”. Never having encountered this utterance before, the learner signals failure and the tutor reveals the answer: SPHERE. The learner’s intention reading process reconstructs a constraint network leading to SPHERE, shown on the top right of the figure. Concretely, this constraint network filters the scene for a single grey object and queries its shape, cor-
rectly resulting in a sphere. Many other constraint networks would also lead to sphere in this scene, e.g. by filtering on a large purple object or a large yellow metal object. However, due to the default best-first search strategy (see Section 2.4.5), the provided constraint program is encountered first by the composer’s search process. Using this constraint program, the learner creates a holophrase construction, pairing the entire question with the entire network. Finally, the learner stores this experience in its memory. The experience consists of the observed question, the scene and the answer revealed by the tutor.

In a later stage of the game, the same question is observed in another scene, marked by ‘Scene 2’ in Figure 6.5. In this scene, the answer is again sphere. The learner is now able to analyse this utterance using its previously acquired holophrase constructions. However, it will not get past the interpretation step as the meaning is not applicable in the scene. Indeed, there is not a single grey object in ‘Scene 2’, but there are multiple, causing the unique operator in the constraint network to invalidate its arguments. To remedy the problem, the learner reconstructs a new constraint network, now taking into account not only the observed question, the tutor’s answer and the current scene, but also its past experiences. As before, many possible networks would lead to sphere in ‘Scene 2’. For example, sphere can be found by filtering on a purple object, filtering on a purple metal object, etc. However, only the network provided on the bottom right of the figure, which filters on a large metal object, satisfies both the current combination of scene and answer and the previously stored combination of scene and answer in the learner’s memory. Hence, this constraint network will be accepted as a valid solution by the composer. Leaving pattern finding out of the equation, the learner can now create a competing holophrase construction.

The observant reader will notice this constraint network still does not correspond to the ground truth constraint network for the question, in particular because the word metal is not represented in the meaning. The learner would need to observe the same question in yet another scene with just the right conditions to revise the constraint network once again. In the meantime, the two holophrase constructions are in competition with each other, which is dealt with by the alignment mechanisms (Section 6.3.8).

Partial Analyses

The composer fully exploits the interplay between intention reading and pattern finding. Specifically, constructions created through pattern finding can provide a partial meaning of a previously unobserved utterance in comprehension. For example, imagine the learner observes the question “Are there any red cubes?” Additionally, imagine the learner has two constructions in its construction inventory: the red-cxn, mapping the form “red” to the meaning (bind color-category ?red red), and the cubes-cxn, mapping the form “cubes” to the meaning (bind shape-category ?cubes cube). When comprehending the utterance, the resulting meaning will consists of these two bind statements. This partial
meaning provides constraints for the composer. In particular, the constraint network being constructed should not only satisfy the communicative goal (i.e. lead to tutor’s intention in terms of a semantic concept), but also match the partial meaning that is provided. A technical description of completing partial programs via the composer is provided in Section 2.4.7. As discussed there, the search process that is performed by the composer during matching is essentially the same, but the information that guides the search is different. Specifically, part of the constraint network is kept fixed while the composer tries to complete it, taking into account the communicative goal and the current scene. Depending on the size of the partial meaning, this drastically reduces the composer’s search space. In the example, only constraint networks containing the two Bind statements will be considered as valid solutions. This allows the composer to rule out a significant number of results during its constraint satisfaction phase. Additionally, this indirectly informs the composer that at least two operators which can link to Bind statements are required, allowing it to skip the evaluation of a large number of composed networks because the matching operation will fail.

While partial meanings typically constrain the composer’s search space, they can also completely prevent the composer from reconstructing a constraint network. This occurs whenever the learner’s pattern finding has made some incorrect generalisation over constructions. As an example, consider an interaction in a very simple scene: two green objects and one blue object. Furthermore, consider the construction green-cxn, which (incorrectly) maps the form “green” to the meaning ( Bind color-category ?x yellow). The tutor, acting as the speaker, might ask the question “How many green objects are there?”. Assuming the learner cannot fully comprehend this utterance, the tutor reveals the answer, namely ‘2’, and the learner tries to reconstruct a constraint network using the aforementioned construction and the provided answer. Specifically, the composer starts from ( Bind color-category ?x yellow) as the initial meaning, and exhaustively searches the space of possible networks, but never finds one that leads to ‘2’ in this particular scene as there are no yellow objects. Nevertheless, from the tutor’s perspective, this is a perfectly valid question. It is only because of the learner’s incorrect form-meaning mapping that intention reading failed. Later on, specifically in Section 6.3.7, I will describe how other repair strategies on the agent’s meta-level can take over in cases like this and allow the agent to learn constructions after all. However, based on this failure during learning, the learner might want to reconsider the construction(s) that caused intention reading to fail, specifically by decreasing their entrenchment scores. This way, intention reading also contributes to overcoming incorrect generalisations made by pattern finding. Additionally, it makes intention reading more efficient, as exhaustive searches like in the example above are avoided.
6.3.6 Pattern Finding with FCG

After intention reading follows pattern finding. The input to pattern finding consists of the observed question and the meaning network reconstructed by intention reading. In order to find patterns, the learner compares this newly created form-meaning mapping against the constructions in its construction inventory, both with respect to form and meaning. The goal is not to create specific constructions for every observed utterance, but to acquire more general constructions that cover multiple utterances, including novel ones. Depending on the applied strategy, the result of pattern finding is one or several new constructions and/or links in the categorial network, both of which can be used for comprehension and production. Three types of constructions are covered by the pattern finding mechanisms: holophrase constructions, lexical constructions and item-based constructions. The categorial network captures the slot-and-filler relations between item-based and lexical constructions.

I discuss four pattern finding strategies that allow to learn constructions through inductive reasoning over differences and similarities between observed questions, reconstructed meanings and existing constructions. The strategies presented here are based on the work by Doumen et al. (forthcoming), where these are applied to a semantically annotated corpus of utterances. Hence, they assume that the intention reading process has already taken place and has reconstructed the correct meaning network. In order to simplify the presentation of the pattern finding strategies in this section, I will make the same assumption. However, it is crucial to note that this assumption does not hold during the experiment, for reasons outlined in Section 6.3.5. Nevertheless, the pattern finding strategies will operate over any form-meaning mapping, as long as they find useful differences and similarities.

Before discussing the four pattern finding strategies, I briefly illustrate the construction application process of holophrase constructions, item-based constructions and lexical constructions. Specifically, I show how item-based constructions and lexical constructions can combine to analyse an utterance and highlight the role of the categorial network (see also Section 2.3.5).

Application of Holophrase Constructions

Holophrase constructions constitute an exact mapping between an utterance and its meaning. Neither the form nor the meaning are decomposed in any way. While such a construction is productive, it only allows to comprehend the exact same utterance or produce the exact same meaning. The application of a holophrase construction, both in comprehension and production, is illustrated in Figure 6.6. Note that the construction is presented in a schematic representation, rather than its actual representation in FCG. At the bottom of the schematic representation, it is indicated whether the construction provides any slots or any arguments that can fill slots. Holophrase constructions, however, provide neither. The comprehension process is shown from top to bottom. The what-is-the-tiny-block-
"What is the tiny block made of?"

Figure 6.6: Construction application of the what-is-the-tiny-block-made-of?-cxn in comprehension (top to bottom) and formulation (bottom to top). Note that the construction is presented in a schematic representation, rather than its actual representation in FCG. A holophrase construction has no slots and does not provide any argument. Figure from Doumen et al. (forthcoming).
MADE-OF?-CXN can only apply if the observed utterance exactly matches the form side of the construction. The result is the meaning side of the construction. The production process is shown from bottom to top. Similarly, the construction is applicable whenever the meaning to express exactly matches the meaning side of the construction, resulting in the form side of the construction.

Application of Item-Based and Lexical Constructions

Figure 6.7: Construction application of the WHAT-IS-THE-?X-BLOCK-MADE-OF?-CXN and the TINY-CXN in comprehension (top to bottom) and formulation (bottom to top). Both constructions collaboratively process the utterance in comprehension or the meaning in formulation. Slots and arguments are tied together via the categorial network on the form side and via unification on the meaning side. Figure from Doumen et al. (forthcoming).

Generalisation over holophrase constructions leads to both item-based and lexical constructions. Item-based constructions are constructions providing one or several slots, while lexical constructions provide arguments for filling those slots. Both slots and arguments
are tied to a particular construction and they are represented in the categorial network. Figure 6.7 illustrates how an item-based construction and a lexical construction collaboratively analyse an utterance in comprehension and express a meaning in production. In particular, the item-based construction what-is-the-?x-block-made-of?-cxn maps between the form pattern ‘What is the ?X block made of?’ and a meaning network that filters the scene for cubes, filters the resulting set for some variable category ‘?category-2’, checks whether the resulting set contains a single element and queries the material of that element. The construction contains an open slot on both the form side, indicated by the yellow ‘?X’ in the figure, and on the meaning side, indicated by the green ‘?category-2’ in the figure. The coupling between these open slots is indicated at the bottom of the construction. The slot specification of the item-based construction states that the open slots can be filled by another construction that provides a mapping between something of category ‘what-is-the-?x-block-made-of(?X)’, which represents the ‘?X’ slot in the form pattern, and something that can unify with ‘?category-2’, i.e. the unbound variable in the construction’s meaning. The lexical tiny-cxn provides just that. It maps the string “tiny” to the bind statement (bind size-category ?category small). The slots of the what-is-the-?x-block-made-of?-cxn can be filled by the arguments of the tiny-cxn on both the form side and on the meaning side. On the form side, the ‘tiny’ category is linked to the ‘what-is-the-?x-block-made-of(?X)’ category in the categorial network. On the meaning side, the ‘?category’ variable of the lexical construction can be unified with the ‘?category-2’ variable of the item-based construction during construction application. As a result, the arguments of the lexical construction are “inserted” in the slots of the item-based construction. This yields a complete meaning network for the utterance in comprehension and a complete utterance for the meaning network in production.

Learning Holophrases

The most basic pattern finding strategy is for learning holophrases. At the start of the experiment, the learner’s construction inventory is empty. When the learner observes its very first utterance, the only thing it can do is to reconstruct its meaning using intention reading, and store an exact mapping between the utterance and the reconstructed meaning. This constitutes a holophrase construction. While the strategy for learning holophrases does not make any generalisations over constructions, the resulting holophrase constructions do form the basis of the learning process. Indeed, because this repair always succeeds, it allows to create many holophrase constructions at the start of the experiment. Later on, other strategies can compare against these holophrase constructions and create generalisations over them.

Generalising over Holophrases

By analysing similarities and differences between a novel observation and an existing holophrase construction, item-based constructions and lexical constructions can be learned.
The former capture the similarities and abstract away over the differences, while the latter capture those differences. Importantly, this pattern finding strategy is restricted to capture minimal differences. In other words, it only copes with a single difference on both the form side and the meaning side. Three variants of this pattern finding strategy exist.

![Diagram](image)

**Figure 6.8:** By analysing similarities and differences between an existing holophrase construction and a new observation, the substitution mechanism allows to learn an item-based construction, two lexical constructions and links in the categorial network. Figure adapted from Doumen et al. (forthcoming).

**Substitution.** The main idea behind this strategy is illustrated in Figure 6.8. The red box represents the learner’s current construction inventory. Among others, it contains the **WHAT-IS-THE-BLOCK-MADE-OF?-CXN**, a holophrase construction mapping the form “What is the block made of?” to a representation of its meaning. For space reasons, the meaning is represented in a schematic way, rather than the complete constraint network. Nodes in this schematic representation can correspond to either primitives or bind statements. A novel utterance “What is the cylinder made of?” shown at the top of the figure, is observed by the learner. As the utterance is novel, its meaning has to be reconstructed through intention reading. The resulting meaning network is shown in the green box\(^2\). The substitution strategy now computes similarities and differences between the observation, i.e. the observed utterance and its reconstructed meaning, and the previously learned holophrase.

---

\(^2\)Remember that, for clarity reasons, it is assumed here that intention reading immediately reconstructs the correct meaning network, but that this is not necessarily the case during the experiment.
construction, both in terms of form and meaning. In this case, it finds a minimal difference on the form side ("cylinder" and "block") and on the meaning side (’F’ and ‘B’). These differences are highlighted in red. This minimal difference on both sides allows to construct four items, shown in the blue box at the bottom of the figure. First, an item-based construction what-is-the-?x-made-of?-cxn is created, capturing the similarities. The differences have been replaced by the slot ‘?X’ on the form side and by a variable ‘?c’ on the meaning side. Next, two lexical constructions can be created, which capture the differences on both form and meaning side. Specifically, the cylinder-cxn maps the form “cylinder” to the meaning ’F’ and the block-cxn maps the form “block” to the meaning ‘B’. Finally, links are made in the categorial network between the slot of the item-based construction and the arguments of both lexical constructions. These links allow the item-based construction and the lexical constructions to work together. Specifically, they allow the grammatical categories of the constructions to match through the categorial network, while the open variables in their respective meaning representations can be unified. Through these links, the generalisation mechanism reveals that ‘cylinder’ and ‘block’ can appear in the same slot and therefore similar grammatical categories.

Figure 6.9: By analysing similarities and differences between an existing holophrase construction and a new observation, the addition mechanism allows to learn an item-based construction, a lexical construction and a link in the categorial network. Figure adapted from Doumen et al. (forthcoming).

Addition. The addition mechanism covers the scenario where an observed utterance and its reconstructed meaning extend a previously acquired holophrase construction by
a single element on both the form side and the meaning side. Specifically, in Figure 6.9, 
the utterance “What is the tiny block made of?” and its reconstructed meaning extend the 
holophrastic what-is-the-block-made-of?-cxn, both on the form side and on the meaning side. The extension is highlighted in red. Here, the learner assumes that it will be possible to extend the holophrase construction in a variety of ways. Therefore, it creates the item-based construction what-is-the-?X-block-made-of?-cxn, the lexical construction tiny-cxn and links in the categorial network between the grammatical categories of these constructions. As before, the item-based construction offers a pattern on the form side tied to an open variable on the meaning side. Together, this constituting a slot. On the form side, this is denoted by ‘?X’ and on the meaning side by ‘?f’. The lexical construction offers arguments to fill the slot. Specifically, some argument of type ‘tiny’ is bound to the variable ‘?g’. Finally, the link in the categorial network ensures that the slot and the argument can be used together during comprehension or production.

Figure 6.10: By analysing similarities and differences between an existing holophrase construction and a new observation, the deletion mechanism allows to learn a holophrase construction, an item-based construction, a lexical construction and a link in the categorial network. Figure adapted from Doumen et al. (forthcoming).

**Deletion.** Generalisation through deletion is the inverse of generalisation through addition. In particular, it handles cases where the observed utterance and its reconstructed meaning are a reduction of a learned holophrase construction by a single element on both the form side and the meaning side. An example is provided in Figure 6.10. The utterance “What is the block made of?” is observed, its entire meaning is reconstructed and
the WHAT-IS-THE-TINY-BLOCK-MADE-OF?-CXN already exists. The differences that trigger the deletion mechanism are again highlighted in red. From this, the deletion mechanism is able to create (i) a holophrase construction WHAT-IS-THE-BLOCK-MADE-OF?-CXN covering the observation, (ii) an item-based construction WHAT-IS-THE-?X-BLOCK-MADE-OF-CXN, (iii) a lexical construction TINY-CXN and (iv) links in the categorial network, associating the slot of the item-based construction to the argument of the lexical construction. Apart from the holophrase construction, these serve to generalise over the already existing holophrase construction that was used as input to the deletion mechanism.

Learning Constructions from Partial Analysis

The third pattern finding strategy allows to learn constructions based on a partial analysis of the observed utterance. This strategy is more tightly integrated with intention reading compared to the previous ones. As discussed, a partial analysis provides constraints on the meaning that is being reconstructed. When intention reading is successful, both the form and the meaning of the constructions providing the partial analysis can be removed from the observed utterance and the reconstructed meaning, respectively. A new construction is created using the remainder of the form and the meaning. Thus, the pattern finding strategy discussed here allows to create new constructions that work together with existing constructions in order to process the entire observation. While the missing elements of the observation from the form side can be extracted from the observed utterance, intention reading is needed to reconstruct the missing elements of the observation from the meaning side. Two scenarios can be identified.

Learning from Lexical Constructions. In the first scenario, one or more lexical constructions provide the partial analysis. An example of such a scenario is given in Figure 6.11. In this example, the learner observes the utterance “What is the red block made of?”. The previously acquired lexical constructions BLOCK-CXN and RED-CXN apply and provide a partial analysis. On the form side, both “red” and “block” are covered and the partial meaning consists of ’B’ and ’F’. Then, intention reading takes over, reconstructing a meaning network for the observed utterance, taking the partial meaning into account. This results in the meaning network shown in the green box. Based on the utterance and the reconstructed meaning, a new item-based construction can be created that incorporates all aspects of the form and the meaning that are not covered by the partial analysis. The former is extracted from the observed utterance, while the latter relies on the intention reading result. In this example, the result is the WHAT-IS-THE-?X-?Y-MADE-OF?-CXN mapping the pattern “What is the ?X ?Y made of?” to a meaning network with two open variables. Note that this item-based construction provides two open slots and that the pattern finding mechanism creates two separate links in the categorial network. The first link allows the RED-CXN to fill the ‘?X’ slot of the item-based construction, while the second link allows the BLOCK-CXN to fill the ’?Y’ slot of the item-based construction.
6.3. METHODOLOGY

“**What is the** red block made of?”

![Diagram of construction inventory and intention reading for red block](image)

**Construction Inventory**

**Intention Reading**

Figure 6.11: From a partial analysis provided by *lexical constructions*, a novel item-based construction is learned and links are added to the categorial network. The observed utterance can now be analysed by combining the existing constructions and the novel construction. Figure adapted from Doumen et al. (forthcoming).

“**What is the** block made of?”

![Diagram of construction inventory and intention reading for block](image)

**Construction Inventory**

**Intention Reading**

Figure 6.12: From a partial analysis provided by an *item-based construction*, a novel lexical construction is learned and links are added to the categorial network. The observed utterance can now be analysed by combining the existing constructions and the novel construction. Figure adapted from Doumen et al. (forthcoming).
Learning from Item-Based Constructions. In the second scenario, an item-based construction provides the partial analysis. An instantiation of this strategy is shown in Figure 6.12. The learner observes the utterance “What is the block made of?” The item-based construction, shown in the red box, almost completely covers this utterance. However, there is no construction that covers the word ‘block’. As in the previous scenario, the partial meaning is passed along to intention reading. A meaning network for the utterance is reconstructed, taking into account the partial meaning, and the meaning of the partial analysis is subtracted. With the remainder, a new lexical construction block-cxn can be created. This construction now covers part of the utterance that was not yet covered and obtained its meaning through intention reading. Specifically, it maps the form “block” to ‘B’. Also, a link in the categorial network is made to tie the slot of the existing item-based construction to the argument of the newly created lexical construction.

This second scenario also handles cases where the partial analysis consists of an item-based construction together with one or several lexical constructions. However, it is only applicable if a single element is missing on both the form side and the meaning side. If multiple elements are missing, e.g. through the application an item-based construction with two slots and no lexical constructions, it would still be possible to derive lexical constructions. However, in such a case, there is referential uncertainty as the learner would need to make hypotheses on which part of the not yet covered form corresponds to which part of the not yet covered meaning. This is currently not explored in this pattern finding strategy.

Extending the Categorial Network

The final pattern finding strategy handles cases where previously acquired item-based and lexical constructions completely cover the observed utterance, but the combination of the item-based slot and the lexical argument(s) has not been observed before. Here, the analysis can be completed by adding all missing links in the categorial network. An example with a single missing link is given in Figure 6.13. In the example, the learner observes the utterance “What is the sphere made of?” The constructions shown in the construction inventory completely cover the utterance but they cannot combine due to a missing link between the ‘what-is-the-?x-made-of?(?X)’ category and the ‘sphere’ category in the categorial network. The repair detects this and adds the missing link, shown on the bottom of the figure. Note that this strategy does not rely on any form of external information, such as the tutor’s feedback, nor any intention reading is required. Hence, the green box in Figure 6.13 is empty.

This pattern finding strategy is also used by the learner in formulation, as it relies solely on information that is already present in the construction inventory. Whenever some combination of item-based and lexical constructions completely covers the meaning network the learner wants to express, but these constructions cannot combine due to missing links in the categorial network, the learner will try to add them. Two additional measures are
6.3. METHODOLOGY

Figure 6.13: When existing item-based and lexical constructions cover the observed utterance, but that slot and filler combination was not observed before, the categorial network needs to be extended. Figure adapted from Doumen et al. (forthcoming).

taken into account when applying this strategy in formulation. First, the learner will not just add any link to the categorial network, but first check whether the grammatical categories are related by looking for a path between them in the categorial network. This avoids situations where an item-based slot that is always filled by a shape, such as the '{X}' slot in the WHAT-IS-THE-{X}-MADE-OF{-CN}, suddenly gets filled by a colour. Second, the link will only be consolidated if the interaction turns out to be successful. By adding new links, the learner is creatively creating new combinations of slots and fillers. However, if this new combination does not lead to communicative success, it will not be tried again in the future.

6.3.7 Competition between Repair Strategies

Intention reading and pattern finding take place on the agent’s meta-level, operationalised through a number of diagnostics and repairs. Each repair implements one of the pattern finding strategies outlined in the previous section and all but one of these strategies requires intention reading. The repairs are in competition with each other, since in many cases it is possible to make several generalisations from the same observation.

A schematic overview of the diagnostics, the repairs and the order in which they are tried is given in Figure 6.14. The learner will always try to complete the interaction through routing processing, shown below the dashed line, before applying meta-level reasoning.
and learning, shown above the dashed line. Both the comprehension and interpretation processes are checked by several diagnostics and each of these diagnostics can trigger multiple repairs. The repairs are tried from bottom to top as outlined in the figure. When a repair produces a fix, routine processing continues with that fix. Otherwise, control is returned to the diagnostic, which then triggers the next repair until one of them succeeds. The 'learn holophrase' repair, which is tried last by all diagnostics, always succeeds. In the following paragraphs, the diagnostics and the repairs are discussed in detail.

![Schematic overview of diagnostics and repairs](image)

Figure 6.14: Schematic overview of diagnostics and repairs. Each diagnostic can trigger multiple repairs. The repairs are tried from bottom to top. When a repair does not produce a fix, the next repair is tried until one of them succeeds. The fix is integrated and routine processing continuous.

**Empty Analysis.** This diagnostic checks whether comprehension resulted in an empty analysis, i.e. when no constructions could apply. The learner first tries to solve this problem by using the repair which generalises over existing holophrase constructions, thereby creating item-based constructions and lexical constructions. If this repair does not produce a fix, the learner simply creates a new holophrase construction, which always succeeds.

**Partial Analyses.** The second diagnostic checks whether comprehension resulted in partial analyses. Here, the learner prefers to solve the problem by creating new links in the categorial network. This requires that the applied constructions completely cover the observed utterance, but lack one or multiple links in the categorial network. If this repair cannot fix the problem, the diagnostic will attempt to apply the repairs which learn new constructions from these partial analyses. Importantly, the learner attempts to learn from every partial analysis that is available, since multiple partial analyses frequently occur in
the elicitation game. To illustrate this, consider an interaction in a scene that does not contain any spheres and where the learner’s construction inventory has the following constructions:

- **CUBE-CXN**: “cube” $\leftrightarrow$ (BIND SHAPE-CATEGORY ?CUBE SPHERE).
- **RED-CXN-1**: “red” $\leftrightarrow$ (BIND COLOR-CATEGORY ?RED GREEN).
- **RED-CXN-2**: “red” $\leftrightarrow$ (BIND COLOR-CATEGORY ?RED RED).

When observing the utterance “What size is the red cube?”, five partial analyses are available: (i) Applying both CUBE-CXN and RED-CXN-1, (ii) applying both CUBE-CXN and RED-CXN-2, (iii) applying only CUBE-CXN, (iv) applying only RED-CXN-1 and (v) applying only RED-CXN-2. The repair strategy considers the partial analyses in the same order as provided during comprehension, namely on the basis of the highest scoring constructions. Assuming the ordering of the partial analyses as provided above, repairs using partial analyses (i), (ii) and (iii) will not yield any fixes because intention reading will not be able to construct any meaning networks that match the provided BIND statements and can be evaluated in the current scene. This is due to the incorrect form-meaning mapping of the CUBE-CXN and the absence of spheres in the scene. Using partial analysis (iv) in the repair might lead to a solution in intention reading, but only if there is a green object in the scene. If there are no green object, only the repair using partial analysis (v) will lead to a result.

When a partial analysis repair cannot produce a fix, the learner can deduce that the constructions used in the repair are incorrect. The learner uses this information to update their entrenchment scores, but only after a successful repair. Specifically, when learning from partial analysis (iv), the learner can decrease the entrenchment score of the CUBE-CXN, as it was unable to reconstruct a meaning network using both the CUBE-CXN and the RED-CXN-1, but successful when using only the RED-CXN-1. Alternatively, when learning from partial analysis (v), the learner can punish the CUBE-CXN and the RED-CXN-1, as both were used in unsuccessful repairs.

To summarise the learner tries all partial analyses that are available, in the same order as they are created during the construction application process, which is based on the scores of the constructions. If a partial analysis repair produces a fix, no other repairs are tried. Constructions used in unsuccessful repairs are punished. If all partial analysis repairs fail, however, all partial analyses are ignored, all applied constructions are punished and the utterance is treated as if it were observed for the first time. Similar to the ‘empty analysis’ diagnostic, the generalisation repair is tried first and the ‘learn holophrase’ repair is only considered afterwards.

**No Answer.** This diagnostic checks whether the interpretation process has failed, i.e. when it was not possible to evaluate the meaning network resulting from comprehension. This can have two possible causes. First, because an incorrect form-meaning mapping was
used during routine processing in comprehension. The resulting meaning network might be nonsensical in the current scene, e.g. trying to identify a yellow object if there are none in the scene. Second, because the ‘extend categorial network’ repair added a link to the categorial network, which also allows the comprehension process to produce a nonsensical meaning. When applied after any of the other repairs, interpretation will always succeed because these repairs all include intention reading. Hence, the meaning network used in interpretation was just now created on the basis of, among others, the current scene.

When this diagnostic is triggered, the utterance is again treated as if it were novel. No information from the construction application process is used. The reasoning behind this is as follows. If a holophrase construction applied and resulted in a nonsensical meaning, the only thing that can be done is to create a new holophrase construction with the same form but a different meaning (or generalise over a similar holophrase, after constructing a new meaning). If, however, an item-based construction and some lexical constructions applied, it is difficult to know which partial information is useful for repairing the problem and which is not. Therefore, the learner does not try to dissect the applied constructions, but immediately learns new ones. As a failed interpretation process is treated as a failed interaction, the entrenchment score of all applied constructions will be lowered anyway.

6.3.8 Alignment

While intention reading and pattern finding are used to acquire new constructions, the goal of alignment is to update the entrenchment scores of constructions such that they are more efficacious for the communicative task. This process corresponds with the concepts of statistical pre-emption (Goldberg, 2011; Boyd and Goldberg, 2011; Goldberg, 2019, Ch. 5) and self-organisation through lateral inhibition (Steels, 1995) (see also Section 2.2.2). I will discuss the alignment mechanism in terms of the latter.

Punishment and Reward

Lateral inhibition rewards and punishes constructions depending on the outcome of the game. If the game fails, the scores of the constructions used during the game are decreased by $\delta_f = 0.4$. Indeed, these constructions were inadequate for solving the communicative task. They should therefore be less entrenched in the agent’s construction inventory and consequently used less often in the future. However, if the game is a success, the constructions that were used become more entrenched. Their score is increased by $\delta_s = 0.1$. A score of 1.0 corresponds to maximal entrenchment. Once reached, the score of a construction is not increased any further.

Competing Constructions

Lateral inhibition regulates competition between constructions. Competing constructions are constructions that could also have contributed to the agent’s comprehension process,
regardless of whether these are more or less abstract than the applied constructions. In comprehension, competing constructions are constructions with the same form, but a different meaning. As other constructions were more suitable for solving the communicative task, the scores of these competing constructions are decreased by $\delta_c = 0.1$.

In the elicitation game, competing constructions are not considered after production. This is due to the template-based design of the questions of the CLEVR dataset, in particular because of its synonymy on the lexical and the grammatical level (see Section 3.3). For example, ‘cube’ and ‘block’ are lexical synonyms, while ‘There is a X; what is its Y?’ and ‘What Y is the X?’ are grammatical synonyms. These are true synonyms in the sense that they are completely interchangeable, regardless of the context in which they are used. Lateral inhibition, on the other hand, steers the agent towards a system with one-to-one mappings, where every form is expressed by a single meaning and vice versa. This is not desirable for the communicative task using this particular dataset. Therefore, lateral inhibition after production only rewards or punishes the constructions that were used during the interaction and ignores competitors with the same meaning, but a different form.

**Removing Constructions and Links**

If a construction reaches a score of 0, it is removed from the construction inventory. At the same time, the associated grammatical categories are removed from the categorial network, together with all of their links.

$\delta$ Values

The exact values by which constructions are rewarded and punished does not influence the global dynamics of the learning process, as long as these values are positive and negative respectively. However, it can influence the speed at which the agents converge to a successful communication system.

### 6.3.9 Tutor Behaviour

While previous sections have focussed on the mechanisms used by the learner, the behaviour of the tutor also affects the learning dynamics. I consider two tutoring strategies, used whenever the tutor acts as the speaker in the elicitation game:

- **Baseline Strategy.** The tutor selects a random scene and a random question. The tutor’s only concern is whether the question actually makes sense in the scene, e.g. not asking about the shape of a brown object when there is no brown object in the scene.

- **Probabilistic Strategy.** The tutor monitors the learner’s success on a per question basis. Specifically, the tutor keeps track of how often the learner has observed each
question and counts how many times each question has led to an unsuccessful game, while scenes are still chosen at random. The tutor chooses between uttering a novel question, sampled randomly, or one that was observed before. In case of the latter, the tutor samples a question according to the probability distribution over the usage and success statistics that it keeps. Specifically, if a particular question resulted in many failed interactions, this question gets a higher probability of being chosen. Indeed, questions leading to failed interactions are more interesting, as they indicate that this part of learner’s construction inventory is not yet sufficiently developed. Using this strategy, the tutor is helping the learner by asking more difficult questions in a variety of different scenes and thereby accelerating the development of the constructions involved.

6.3.10 Learning Dynamics

The setup of the elicitation game and the learning mechanisms outlined in Sections 6.3.1 to 6.3.8 enable the learner to bootstrap an effective and efficient communication system in the form of a construction grammar. The learning dynamics of the elicitation game can be summarised as follows.

Whenever the learner cannot complete the interaction acting as the listener, the tutor reveals the answer to the question it asked. This is effectively the tutor’s intention, as the tutor wanted the learner to come up with a network in comprehension, use it in interpretation and ultimately end up with that answer in mind. Meta-level diagnostics and repairs become active in order to expand the learner’s construction inventory. In particular, using intention reading, the learner hypothesises about a semantic network that leads to the provided answer in the current scene. The reconstruction of meaning is a huge search problem. Therefore, the learner relies on multiple sources of information, namely the observed utterance, the current scene, past experiences, known constructions and the answer to the question given by the tutor. Nevertheless, the reconstructed meaning might be incorrect. Afterwards, the reconstructed meaning is paired with the observed utterance. Pattern finding is used to look for differences and similarities between this newly created form-meaning pairing and previously acquired constructions. Different pattern finding strategies are implemented for learning either holophrase constructions, item-based constructions, or lexical constructions. Simultaneously, a network of grammatical categories modelling slot-and-filler relations emerges. The cognitive capacities of intention reading and pattern finding are tightly interwoven. Specifically, intention reading relies on the partial analyses provided by previously acquired constructions (if any) to constrain its search space, while pattern finding relies on the result of intention reading to create constructions and compositional generalisations over them. The alignment dynamics create a positive feedback loop between the use and success of constructions. This ensures that constructions that can be used successfully become highly entrenched, while unsuccessful and competing constructions become less entrenched and eventually disappear. The
alignment mechanism thus enables the agent to eventually remove constructions with incorrectly reconstructed meanings since these cannot be used to successfully complete the communicative task. The memory mechanism added to intention reading allows to reconstruct a different, more accurate meaning, taking past experiences into account.

Importantly, the presented learning mechanisms do not posit a build-in bias towards more abstract constructions. However, more abstract constructions are inherently applicable in a wider range of situations and will therefore be used more frequently. Due to the alignment dynamics, this will result in higher entrenched scores for more abstract constructions and in lower scores and generally fewer less abstract constructions. On the contrary, the learner will only learn new constructions when it cannot complete the game with its current construction inventory. Hence, after a series of elicitation games, the learner’s construction inventory will consist of a blend of abstract and less abstract constructions, specifically tailored for the communicative task at hand.

6.4 Experiments

This section presents the experimental results obtained in the elicitation game. First, in Section 6.4.1, I describe a number of metrics for evaluating the learner and comparing different strategies. Next, Section 6.4.2 presents the main results in terms of these metrics. By default, the tutor uses the baseline strategy (see Section 6.3.9). However, in section 6.4.3, I present the effects of the probabilistic tutoring strategy. All experiments are implemented using the Babel software package (see Section 2.2.4). Unless otherwise specified, the presented results are based on ten independent runs of 250,000 interactions each. The filled areas around the lines on the plots represent the 5th and 95th percentile.

6.4.1 Metrics

The learning process during the elicitation game is evaluated through four quantitative metrics that were introduced by Doumen et al. (forthcoming) and one additional metric introduced here:

- **Communicative success** over time is computed by comparing the listener’s answer to the speaker’s intended topic. If both the speaker and the listener could complete the interaction through routine processing and the listener’s answer is correct, the interaction is assigned a value of 1. In all other cases, the interaction is assigned a 0. Thus, if the learner requires meta-layer reasoning and learning to complete the game, if the listener’s answer is incorrect, or if the listener fails to provide an answer the interaction has failed. There is a single exception to this scheme, which is the repair strategy that extends the categorial network. If this repair was used, the interaction can still be counted as a success if the answer is correct because this particular repair strategy does not require any external information or feedback. The
CHAPTER 6. LEARNING STRUCTURES

1’s and 0’s returned by the communicative success metric are plotted using a sliding window of 100 interactions.

- **Grammar size** over time is computed by counting the number of constructions in the learner’s construction inventory after every interaction.

- **Number of constructions per type** over time also counts the constructions in the learner’s inventory, but divides them into three groups: holophrase constructions, lexical constructions and item-based constructions.

- **Active repair strategies** over time keeps track for each of the repair strategies whether it was used in the current interaction. This is recorded as a 1, if it was active, or a 0, if it was not active. These values are plotted using a sliding window of 250 interactions.

- **Grammatical categories and links** over time keeps track of the number of nodes and the number of edges in the learner’s categorial network.

### 6.4.2 Main Results

The metrics just introduced are presented for the elicitation game in Figures 6.15 to 6.19.

**Communicative Success**

Figure 6.15 shows the communicative success (solid teal line on the left y-axis) and the grammar size (dashed yellow line on the right y-axis) as a function of the number of games played (x-axis). Figure 6.15a shows all 250,000 interactions, whereas Figure 6.15b zooms in on the first 25,000 interactions.

Both the communicative success and the grammar size start at 0, as the learner starts with an empty construction inventory. However, the communicative success rises quickly, with 78.7% of the interactions being successful after only 5,000 interactions, rising to 98.9% after 10,000 interactions. This is a remarkable achievement as the learner only could have observed half of the 10,044 unique questions that can be uttered by the tutor. Indeed, in every interaction, the discourse roles of the agents are randomly decided with equal probability and thus, in only half of the interactions is the tutor chosen as the speaker, allowing the learner to observe some utterance. Even more, as the utterances are randomly sampled by the tutor, there is no guarantee that the tutor will utter a novel utterance when it is the speaker.

After 25,000 interactions, when the learner has most likely observed all possible utterances, the communicative success is over 99.9% and it remains so until the end of the experiment. Depending on where exactly the sliding window of the communicative success is computed, the success does not completely converge to 100% after 25,000 interactions. This is because it takes a large number of interactions for all incorrect hypotheses (i.e. incorrect
constructions) to be removed from the construction inventory, which occasionally causes an interaction to fail. This is due to the tutor’s baseline strategy, presenting utterances in random order. In order to remove some incorrect construction, the learner would need to observe a relevant utterance in just the right scene such that the interaction fails and the construction is punished. Even more, this would need to occur several times such that the incorrect construction is punished enough times and its score reaches zero such that it gets removed from the construction inventory, without any successful interactions using that construction in the meantime.

(a) Overview of all 250,000 interactions.  
(b) Zooming in on the first 25,000 interactions.

Figure 6.15: Evolution of the communicative success (solid teal line on left y-axis) and the grammar size (dashed yellow line on right y-axis). Both metrics rise very quickly. Communicative success rapidly converges, while it does take a very long time to get rid of all incorrect form-meaning mappings, as indicated by the much slower decline in grammar size.

**Grammar Size**

The evolution of the grammar size shown in Figure 6.15 clearly shows that it takes a large number of interactions to converge to a stable construction inventory. At first, there is an explosion of constructions. After 5,000 interactions, on average 1,048 constructions were learned. At this point, the construction inventory contains many competing constructions. For example, holophrase constructions or lexical constructions with the same form, or item-based constructions with form patterns that subsume one another. Due to the alignment dynamics, the construction inventory gradually reduces in size, reaching on average 491.9 constructions after 25,000 interactions. The rate of decrease is rapid at first but decreases as the experiment progresses. The construction inventory reaches on average 149.3 constructions after 250,000 interactions. It is important to note that it is not the goal to learn one particular set of constructions. For example, the learner will not be evaluated on whether its construction inventory is identical, or even has the same number of constructions as the tutor’s inventory. Instead, the goal of the learner is to be successful at the communicative task and learn an efficient construction inventory for doing so. This is exactly what the learner achieves, as shown by the metrics in Figure 6.15.
Number of Constructions per Type

![Graphs showing evolution of constructions over time](image)

(a) Overview of all 250,000 interactions  
(b) Zooming in on the first 10,000 interactions

Figure 6.16: Evolution over time of the number of construction per type, including holophrase constructions (solid teal line), item-based constructions (dashed yellow line) and lexical constructions (dotted red line).

Figure 6.16 breaks down the evolution of the grammar size per construction type. A complete picture is provided in Figure 6.16a, while Figure 6.16b zooms in on the first 10,000 interactions. A distinction is made between holophrase constructions (solid teal line), lexical constructions (dashed yellow line) and item-based constructions (dotted red line).

At the start of the experiment, only holophrase constructions are learned. Soon after, the learner is able to generalise over them and create both item-based and lexical constructions. There is an abundance of item-based constructions, with as many as 837.6 constructions on average after 5,000 interactions. As more general item-based constructions are applicable in a wider number of cases, they become dominant and take over their less abstract competitors. This is clearly seen in Figure 6.16a by the rapid decline of item-based construction from interaction 5,000 onwards. At the end of the experiment, after 250,000 interactions, there are on average 85.2 item-based constructions remaining.

There is less competition among the lexicon constructions, steadily climbing to 26.4 constructions after 5,000 interactions and reaching on average 32.5 constructions after 10,000 interactions. This number only increases slightly, to 32.8 constructions on average, at the end of the experiment. The theoretical limit of 35 lexical constructions was reached in four out of ten experimental runs. In these four cases, there are generally fewer item-based constructions (67.8 compared to 96.8 in the other six runs) because the additional lexical constructions allow for more generalisations. I argue that the theoretical limit of 35 lexical constructions is not consistently achieved due to the interplay between the (random) order in which questions are observed and the restriction to minimal differences employed by some of the pattern finding strategies. I discuss this matter more in depth in light of the experimental results concerning the grammatical categories and links.

The number of holophrase constructions keeps rising for the first 5,000 interactions, reach-
ing 184 on average. Afterwards, however, there is a slow decline in holophrase constructions as the agent learns to combine item-based constructions and lexical constructions to cover the same utterances. Nevertheless, the holophrase construction have not yet completely disappeared after 250,000 interactions. There are still on average 31.3 holophrase constructions remaining.

**Active Repair Strategies**

![Graph a](image1.png) ![Graph b](image2.png)

(a) Overview of all 250,000 interactions  
(b) Zooming in on the first 25,000 interactions

Figure 6.17: Active repair strategies over time. Note that the y-axis only goes up to 0.5 because most repairs are applied when the learner acts as the listener, which is in half of the interactions.

Figure 6.17 presents the active repair strategies over time. Figure 6.17b focusses on the first 25,000 interactions. Note that the y-axis in this figure goes up to 0.5 since most repairs are active when the learner acts as the listener, which is in half of the interactions. The repair strategy that generalises over holophrases (dashed yellow line) contains the substitution variant, the addition variant and the deletion variant. The partial analysis repair strategy was split up, depending on whether the partial analysis contained at least an item-based construction (dark-blue dash-dotted line) or only lexical construction(s) (dotted red line).

The figure clearly highlights several stages during learning. At first, the repair strategy for learning holophrases is most active. With only a few holophrases, the learner is already able to generalise over some of them. This results in a spike for the partial analysis repair using lexical constructions. The few lexical constructions that are learned at this point seem to appear in many newly observed utterances, allowing the agent to use them in partial analyses. During this spike, the repair strategies for learning holophrases and generalising over holophrases are also still active, but they are in decline. These repair strategies now only apply to utterances that do not contain any of the already learned lexical constructions. Together with the holophrase generalisation repair, the partial analysis repair strategy with lexical constructions causes item-based constructions to flourish, as was seen in previous figures. This now allows for a new stage in learning where the partial analysis repair with item-based constructions and the repair extending the categorial
network become more active. Once enough item-based constructions and lexical constructions are in place, due to the other repairs, the repair for extending the categorial network becomes most active and is sufficient to handle all problems diagnosed by the learner. In other words, sufficient item-based and lexical constructions where learned, but now it is only a matter of learning which fillers can be used in which slots. After 25,000 interactions, nearly all interactions are handled by routine processing. Either one of the partial analysis repairs or the repair for extending the categorial network sporadically apply. In case of the former, this is due to an incorrect construction that took a large number of interactions to be removed from the linguistic inventory, as discussed before. Once this happens, new construction(s) need to be learned to take its place and cover the observed utterance. For the latter repair strategy, this is due to the learner acting as the speaker and exploring a new slot-filler combination, or due to a previously unobserved slot-filler combination that occurs in comprehension. In the second case, this does not necessarily mean that the utterance triggering the repair was observed for the first time. Instead, it can also mean that some holophrase construction or less abstract item-based construction was just removed from the construction inventory and a more abstract item-based construction has taken over its role. The slot-filler combination that was previously covered by other construction(s) now also needs to be covered by this more abstract construction, hence requiring the repair strategy for extending the categorial network.

Grammatical categories and links

Figure 6.18: Evolution over time of the number of nodes (solid teal line, left y-axis) and edges (dashed yellow line, right y-axis) in the categorial network.

The categorial network captures the slot-and-filler relations between item-based constructions and lexical constructions. It is gradually built up during the experiment through the application of the repair strategies. Entrenchment scores on these relations are incremented after a successful comprehension or production process. Figure 6.18 shows the
evolution of the number of nodes and the number of edges in the categorial network over the course of 250,000 interactions. Near the start of the experiment, there is an explosion of nodes, which is consistent with the previously discussed results. Consequently, the number of edges also rises quickly. Over time, the number of nodes decreases as the number of item-based constructions goes down. The links in the network also decrease, but to a lesser extent as more slot and filler relations are discovered. This is indicated by the spike of the repair for extending the categorial network in Figure 6.17. At the end of the experiment, there are on average 213.1 nodes and 1164.9 links between them.

A fragment of the categorial network resulting from one experimental run is shown in Figure 6.19. The entrenchment scores were omitted from the figure for readability. Nodes which have links to many common nodes are drawn such that they are closer together. Thereby, this figure highlights the similarity between grammatical categories that has emerged during the experiment and is captured through the distribution of links between nodes. For instance, on the top left, there is a cluster of grammatical categories which captures plural nouns depicting the shape of objects. For example, the argument provided by the spheres-cxn can be used to fill three slots in three different item-based constructions. These are the ‘?x’ slot in the are-any-?y-?x-visible-cxn, the ‘?x’ slot in the what-number-of-?a-?z-?y-?x-are-there-cxn and the ‘?x’ slot in the how-many-?x-are-there-cxn. Other arguments that are compatible with all three aforementioned ‘?x’ slots are those provided by the objects-cxn, the balls-cxn, etc.

Even within a cluster of nodes, there can still be subtle differences. For example, the cluster on the top right captures the grammatical categories for conceptual categories. However, the argument provided by the shape-cxn seems to be compatible with a number of different slots compared to the arguments of the material-cxn, color-cxn and size-cxn. A similar pattern is in found in the cluster in the middle of the figure for the arguments of the thing-cxn and object-cxn as compared to the arguments of the ball-cxn, cube-cxn, sphere-cxn, cylinder-cxn and the block-cxn.

The cluster on the bottom of Figure 6.19 captures adjectives used to describe properties of objects from the CLEVR dataset, i.e. material, colour and size. These are grouped together in the same cluster of grammatical categories as many slots of item-based constructions are compatible with multiple of the arguments provided by these types of constructions. For example, the ‘?x’ slot in the is-there-a-?x-?y-cxn is compatible with, among others, the arguments of the shiny-cxn, the red-cxn and the large-cxn.

The four clusters shown in Figure 6.19 have emerged in four out of ten experimental runs. This corresponds with the theoretical maximum of 35 lexical constructions that were also learned in four out of ten runs. In the other six runs, it is always the cluster of grammatical categories capturing conceptual categories (top right) that is missing. I argue that this is due to the interplay between the random order in which questions are observed and the restriction to minimal differences employed by some of the pattern finding
Figure 6.19: Fragment of the categorial network constructed throughout the experiment. Entrenchment scores on links have been left out for readability. Four distinct clusters of grammatical categories have emerged.
strategies. Because of this restriction, generalisations that capture conceptual categories would need to be learned through pattern finding over holophrase constructions of the kind \texttt{WHAT-COLOR-IS-THE-<NP>-cxn} and \texttt{WHAT-SIZE-IS-THE-<NP>-cxn}, where \texttt{<NP>} is a placeholder for a fully-instantiated, variable-length noun phrase such as ‘green cube’, ‘large metal sphere’, ‘tiny purple rubber block’, etc. However, due to the design of the CLEVR questions, holophrase constructions which have a minimal difference concerning the shape, colour, size or material of objects and have identical conceptual categories are more common than holophrase constructions which have a minimal difference concerning the conceptual categories and have identical shapes, colours, sizes and materials. Therefore, in four out of six experimental runs, the agent can more rapidly acquire item-based constructions of the kind \texttt{WHAT-COLOR-IS-THE-<?X>-cxn}, where \texttt{<?X>} is a placeholder for a variable number of slots, compared to an item-based construction of the kind \texttt{WHAT-?X-IS-THE-<NP>-cxn}. After learning such a construction, it becomes more difficult to learn a \texttt{COLOR-cxn} due to the design of the pattern finding strategies and specifically the lack of strategies that can generalise over item-based constructions. Additionally, there are more opportunities for learning generalisations over shapes, colors, sizes and materials as compared to conceptual categories since the former occur in all types of questions, while the latter are only part of the type of questions for querying attributes of objects. Lifting this minimal difference restriction is crucial for further scaling the pattern finding strategies and is part of future work.

6.4.3 Probabilistic Tutor Strategy

The main differences between the tutor’s baseline strategy and the probabilistic strategy are illustrated in Figure 6.20. In this figure, the tutoring strategies are compared across the metrics introduced in Section 6.4.1.

Figure 6.20a, which compares the communicative success, shows that the agents are more successful in communicating early on in the experiment when the tutor uses its probabilistic strategy. For instance, after 5,000 interactions, there is communicative success in 78.7% of the interactions using the baseline strategy as compared to 85.7% when using the probabilistic strategy. However, after 10,000 interactions, this difference has disappeared and the agents reach a stable level of communicative success. From interaction 25,000 onwards, the communicative success does not drop below 99.9%.

Figures 6.20b to 6.20e compare the tutoring strategies with respect to the number of constructions, either aggregate or per type. The probabilistic tutor strategy allows the learner to converge to a stable construction inventory more rapidly. Additionally, there is less competition between constructions during learning, indicated by the smaller peak at the start of the experiment in Figure 6.20b. In other words, by using the probabilistic tutor strategy, the tutor is helping the learner to remove incorrect constructions more rapidly from its construction inventory. This is easily explained by the characteristics of this tu-
CHAPTER 6. LEARNING STRUCTURES

Figure 6.20: Comparing the tutor’s baseline strategy (solid teal lines) to the probabilistic strategy (dashed yellow lines) across several metrics. The probabilistic strategy allows for faster convergence and more abstraction, resulting in fewer constructions.
toring strategy. Specifically, the tutor keeps track of which utterances are difficult for the learner (i.e. often lead to unsuccessful interactions) and repeats these utterances more often. This allows the learner to acquire one or several constructions that can correctly handle those utterances. Once the correct constructions have been learned, the tutor’s statistics are still trailing behind. These utterances, which can now be handled successfully, will still be observed several times before the probability for sampling this question decreases. This allows the newly acquired constructions to be used several times and become strongly entrenched. Additionally, this allows the competing constructions to be removed more rapidly from the construction inventory, thereby explaining the more rapid decrease in grammar size.

Table 6.2: Comparing the number of constructions learned after 250,000 interactions across both tutor strategies.

<table>
<thead>
<tr>
<th></th>
<th>Holophrase</th>
<th>Lexical</th>
<th>Item-Based</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>31.3</td>
<td>32.8</td>
<td>85.2</td>
<td>149.3</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>24.5 (-6.8)</td>
<td>34.0 (+1.2)</td>
<td>74.6 (-10.6)</td>
<td>133.1 (-16.2)</td>
</tr>
</tbody>
</table>

The probabilistic tutor strategy allows for slightly more generalisations over constructions. This is illustrated through Table 6.2, comparing the number of constructions at the end of 250,000 interactions across both strategies. On average, the learner is able to acquire slightly more lexical constructions. In seven out of ten experimental runs, the theoretical maximum of 35 lexical constructions was learned. For the baseline strategy, this was in four out of ten runs. Together with the competing constructions being removed more efficiently from the construction inventory, the larger amount of lexical constructions also allows for fewer item-based constructions, as these will be more abstract and cover a wider range of utterances. However, it is important to note that the total number of constructions, as indicated in Table 6.2, is not the end goal of the experiment. Rather, it is about the evolution of the size of the construction inventory and being able to successfully complete the communicative task between agents.

### 6.5 Conclusion

In this chapter, I have presented the third and final experiment of this dissertation. The theoretical and empirical foundations of this experiment can be found in usage-based theories of language acquisition (Section 6.2.1). Specifically, the cognitive capacities of intention reading and pattern finding, as introduced by Tomasello (2003, 2009b), are central in this chapter. Both intention reading and pattern finding have been investigated through several computational models (Section 6.2.2). The experiment presented in this chapter pushes forward the state of the art in three ways. First, it not only investigates intention
reading and pattern finding separately, but focuses on how these processes jointly facilitate language acquisition. Second, the experiment takes place on a much larger scale than previous work. Third, the agent relies on much fewer scaffolds as compared to previous work. Crucially, the goal of the experiment is not to construct a realistic computational model of how children acquire language through intention reading and pattern finding. Instead, the goal is to construct a mechanistic model of these cognitive capacities such that they can be used by an intelligent, autonomous agent in order to acquire a communication system in the form of a construction grammar. In the experiment, as described in Section 6.3, intention reading and pattern finding have been implemented through IRL and FCG, respectively. Concretely, intention reading consists of composing primitive cognitive operators, provided to the agent, into semantic networks that allows to infer the interlocutor’s intentions. Pattern finding consists of different strategies for analysing differences and similarities across form and meaning, and thereby learn compositional generalisations over constructions. This extends earlier work by Doumen et al. (forthcoming). Together with alignment dynamics, intention reading and pattern finding are provided as meta-level learning operators. Together with a tutor agent, who has an established communication system, the learner agent plays the elicitation game. This game models an interactive version of the visual question answering task and relies on data from the CLEVR dataset. Over the course of many such games, the meta-level learning operators allow the agent to gradually shape its construction inventory such that it becomes efficacious for the communicative task. The agents’ communication is evaluated through a number of quantitative metrics (Section 6.4). These convincingly show that the agent is able to gradually shape an effective and efficient communication system, in the form of a construction grammar, together with an emergent network of grammatical categories. The grammar can be used successfully for both language comprehension and production in order to solve the communicative task of visual question answering.

6.5.1 Contributions

This experiment directly contributes to the primary objective of this dissertation (O1). Specifically, it introduces adequate representations and learning mechanisms in order to acquire both the morpho-syntactic and semantic structures underlying linguistic utterances. The representations used by the agents are constructions, a network of grammatical categories and procedural representations of meaning. The learning mechanisms take the form of repair strategies, acting on the agent’s meta-level and intervening whenever the agent’s current construction inventory is unable to complete a comprehension or production task. They are based on the cognitive capacities of intention reading and pattern finding (Tomasello, 2003, 2009b) and on alignment dynamics (Garrod and Doherty, 1994; Pickering and Garrod, 2006). Together, these representations and learning mechanisms allow to agent to acquire a productive communication system that can be used to solve a challenging communicative task (O2), in this case visual question answering on the CLEVR
6.5. CONCLUSION

dataset.

The contribution and potential impact of this chapter spans five areas:

- **Evolutionary Linguistics** The origins, emergence, evolution and acquisition of grammar is a focal point of the research within the language game paradigm. Prior language game experiments have studied the constructivist co-acquisition of syntax and semantics (Gerasymova and Spranger, 2010; Beuls et al., 2010; Spranger and Steels, 2015). However, the experiment presented in this chapter advances the state of the art in two ways. First, it operates on a much larger scale, using utterances of considerable complexity both in terms of morpho-syntax and semantics, and it does not focus on a specific linguistic phenomenon, such as the Russian aspectual system (Gerasymova and Spranger, 2010), the Hungarian agreement system (Beuls et al., 2010) or English spatial language (Spranger and Steels, 2015). Second, there are fewer scaffolds in the experiment compared to prior work. For instance, the agent does not receive a predefined lexicon as in Beuls et al. (2010) or a taxonomy of semantic concepts that guides the generalisation process of constructions as in Spranger and Steels (2015). In this experiment, the agent only receives a number of primitive cognitive operators and a repertoire of semantic concepts. The latter are not used to steer generalisation processes and could also be acquired through the methodologies outlined in Chapter 4 or Chapter 5.

- **Usage-based Language Acquisition.** Similar to the work by Doumen et al. (forthcoming), the experiment presented in this chapter provides computational evidence for the cognitive plausibility of theories from usage-based language acquisition, in particular Tomasello (2003)’s intention reading and pattern finding. By operationalising these cognitive capacities and their interplay in an agent-based simulation, learning dynamics that are similar to those observed in the psycholinguistics literature (i.a. Pine and Lieven (1997); Tomasello (2003); Ambridge and Lieven (2015)) have been revealed. Specifically, starting out with holistic mappings between form and meaning, the agent learns to generalise over them compositionally, resulting in more abstract item-based and lexical constructions, along with a network of grammatical categories that captures the distribution of item-based slots and their lexical fillers.

- **Construction Grammar.** Also similar to Doumen et al. (forthcoming), the experiment supports the theoretical underpinnings of the field of construction grammar (see Section 2.3.1 for the basic tenets). It provides a computational account on the representation, processing and learning of form-meaning mappings, or constructions. These constructions capture all linguistic knowledge of the agents in a dynamic system that gradually becomes conventionalised through an entrenchment process that corresponds with statistical pre-emption (Goldberg, 2011) or self-organisation through lateral inhibition (Steels, 1995). The constructions are all sit-
uated on the lexicon-grammar continuum, ranging from fully concrete constructions, e.g. holophrase constructions, to more abstract constructions, e.g. item-based constructions. The achieved degree of abstraction gives a unique insight into the compositional and non-compositional aspects of the learned language through the pattern finding strategies. Next to constructions, a network of construction-specific and functionally motivated grammatical categories emerges, in the spirit of Croft (2001)’s radical construction grammar.

• **Visual Question Answering.** The elicitation game models an interactive version of the task of visual question answering. As discussed in Section 5.2, VQA is typically tackled in one of three ways: (i) monolithic approaches (i.a. Gao et al. (2015); Ren et al. (2015); Ma et al. (2016)), (ii) modular neural network approaches (i.a. Andreas et al. (2016b); Johnson et al. (2017b); Hu et al. (2017a, 2018); Mascharka et al. (2018); Bahdanau et al. (2019)) or (iii) neuro-symbolic approaches (i.a. Yi et al. (2018); Mao et al. (2019); Han et al. (2019)). All of these approaches rely on huge amounts of training data. The first approach is easily deceived by statistical biases or shows poor generalisation, as shown by Agrawel et al. (2016); Goyal et al. (2017); Manjunatha et al. (2019); Das et al. (2019). The second approach typically overcomes these issues but often requires semantically annotated questions in order to train a model to perform semantic analysis on the question. This data is not always available. Finally, black-box architectures are used in most of the aforementioned models, making it unclear how and why a particular answer was given. The methodology presented in this chapter stands in stark contrast with these approaches to VQA. Similar to the monolithic approaches, the agent is only presented with scenes, questions and their answers. No examples of semantically annotated questions need to be provided since the agent autonomously discovers the actions required to compute the answer through a process of meaning creation. Furthermore, the presented methodology is much more data-efficient, as very high levels of success are achieved after a single epoch (i.e. observing every possible question just once), and the agent’s representations and reasoning processes are completely transparent and human-interpretable. Finally, the agent’s communication system is completely open-ended and operates bidirectionally using the same representations and processing mechanisms, an achievement that cannot be matched by the previously cited works.

• **Intelligent Systems.** Most importantly, this chapter pushes forward the state of the art in the development of autonomous, intelligent agents with human-like communication systems (Mikolov et al., 2016; Shah et al., 2019). In particular, this chapter introduces a novel methodology that allows an intelligent agent to acquire an inventory of constructions that is suitable for solving a communicative task through situated communicative interactions with *indirect supervision only*. Indeed, the agent never directly observes the exact form-meaning mappings that should be learned nor the meaning underlying any of the utterances. Given only utterances, feedback
on their underlying intentions and a collection of primitive cognitive operators, the agent engages in a highly non-trivial process of meaning creation, i.e. intention reading, and combines this with a process of form-meaning abstraction, i.e. pattern finding. Cooperatively, these processes allow to bootstrap a successful communication system. The presented methodology is completely transparent, both in terms of the applied learning strategies and the resulting inventory of constructions. Through the online and incremental learning approach, the agent acquires useful linguistic knowledge even after a single interaction. Moreover, as there is no separation between a training phase and an operational phase, the alignment dynamics enable the agent to remain ever-adaptive, for example when the environment or the communicative task should change. This chapter thus paves the way for autonomous, intelligent agents that incrementally learn a human-like communication system in the form of a large-scale, open-ended grammar that facilitates both language comprehension and production and solves a particular communicative task, in this case visual question answering.

Together with the previous chapters, this chapter completes the major objectives of this dissertation in that it provides adequate representations and learning mechanisms that enable autonomous agents to acquire linguistic structures on the conceptual, morphosyntactic and semantic level that are suitable for solving communicative tasks in their environment and that bring to bear the key desirable properties of human languages, such as robustness, flexibility, adaptivity, learning efficiency and expressivity. These representations and learning mechanisms have been validated on challenging communicative tasks in concrete environments, specifically the task of visual questions answering on the CLEVR dataset (Johnson et al., 2017a). Apart from completing this objective, this chapter also ties together all previous chapters. Specifically, it offers a methodology for learning constructions through situated interactions, instead of manually designing them, as was done for the CLEVR grammar (Chapter 3). A collection of semantic concepts was provided here as a scaffold, but these can also be learned from the environment through discrimination (Chapter 4). Alternatively, instead of using the agent’s perception which operates on the symbolic level, it could also operate on both the symbolic and the sub-symbolic level using hybrid procedural semantics (Chapter 5).
Chapter 7

Conclusions

7.1 Introduction
The primary objective of this dissertation, as laid out in Chapter 1, was to introduce novel representations and learning mechanisms that enable autonomous agents to acquire linguistic structures on the conceptual, morpho-syntactic and semantic level, that are suitable for solving communicative tasks in the agents’ environment, and that bring to bear the key desirable properties of human languages, such as robustness, flexibility, adaptivity, learn-
ing efficiency and expressiveness. The representation, processing and learning of linguistic structures on the conceptual level allows agents to link their low-level sensorimotor experiences to higher-level symbolic concepts that are meaningful in the environment and for the communicative task and can be used for reasoning, while structures on the morphosyntactic and semantic level allow agents to convey information on how these concepts interact and how they can be used to dampen referential ambiguities through grammar.

The presented representations and learning mechanisms are intended to advance the state of the art in the development of autonomous agents with human-like communication systems. Prior work in this area was either limited in terms of the complexity of the developed communication systems (Doumen et al., forthcoming) (i.a. Dominey (2006); Chang (2008); Gaspers and Cimiano (2014); Abend et al. (2017)), applied to specific linguistic phenomena (Doumen et al., forthcoming) (i.a. Gerasymova and Spranger (2012); Spranger (2017); Beuls et al. (2010)), or had difficulties in capturing the conditions in which human languages emerge and evolve (i.a. Das et al. (2017b); Foerster et al. (2016); Lazaridou et al. (2016b); Mordatch and Abbeel (2018)), which has important repercussions on the emerged languages (Van Eecke and Beuls, 2020). In contrast, the novel representations and learning mechanisms presented in this dissertation operate on a larger scale compared to previous work on learning communication systems through task-based, situated interactions, both in terms of the complexity of the input and the linguistic phenomena considered. Additionally, they focus on the key desirable properties of human communication systems, and, from a computational point of view, are designed to be transparent and human-interpretable. In turn, the developments in agent-based human-like communication systems allow for more advanced experiments on the emergence and evolution of languages, especially in the direction of grammars, and facilitate the development of more capable intelligent autonomous agents that can interact among themselves or with humans through natural language.

In Chapter 2, I have situated the primary objective within the cultural perspective on language evolution, and more specifically within the language game paradigm (Steels, 1995). This paradigm tackles the question of how linguistic conventions can emerge through local interactions and coordination in a population of autonomous agents that are situated in their native environment. I have presented constructions, i.e. conventionalised form-meaning mappings, and procedural cognitive semantics, i.e. algorithmically executable representations of meaning, as two representations that play a central role in achieving the objectives of this dissertation. I introduced Fluid Construction Grammar (FCG) and Incremental Recruitment Language (IRL) as powerful formalisms that allow to computationally represent, process and learn constructions and procedural semantic representations, respectively. By exploiting principles from evolutionary systems, that are well known for solving problems in a manner that is robust, flexible and adaptive to the environment, the language game paradigm is aimed at learning human-like communication systems that exhibit the same key properties. Because of their tight integration within this paradigm
and by operationalising insights from linguistics and cognition, both FCG and IRL focus on bidirectional processing on the conceptual and linguistic level that is robust, flexible, adaptive, efficient and expressive.

The secondary objective of this dissertation was to validate the novel representations and learning mechanisms through case studies that tackle challenging communicative tasks in concrete environments. In Chapter 3, this objective was anchored in the task of visual question answering, and more specifically using the CLEVR benchmark dataset (Johnson et al., 2017a). Answering natural language questions about images requires perceptual, linguistic and reasoning abilities. By placing both the linguistic and reasoning abilities at the forefront, the CLEVR benchmark task is well suited for the purposes of this dissertation.

The representations and learning mechanisms for conceptual, morpho-syntactic and semantic structures introduced in Chapters 4, 5 and 6 are concretely operationalised within the language game paradigm and tightly integrated with FCG and IRL. By tackling various aspects of the CLEVR benchmark task, I have shown that the introduced representations and learning mechanisms allow autonomous agents to acquire communication systems with which they can solve the visual question answering task and that exhibit many of the same properties as found in human languages.

In the remainder of this chapter, I present an overview of the achievements of this dissertation in Section 7.2. Afterwards, in Section 7.3, I discuss a number of limitations of the presented contributions and the avenues of future research that can be pursued by addressing these limitations.

7.2 Achievements

The two objectives of this dissertation have materialised into five concrete achievements. These are (i) a large-scale computational construction grammar for solving the CLEVR benchmark task (Section 7.2.1), (ii) a methodology for learning grounded concepts through discrimination (Section 7.2.2), (iii) a fully explainable grounded language processing system applied to visual question answering (Section 7.2.3), (iv) hybrid procedural semantics (Section 7.2.4) and, for the most important contribution, (v) a mechanistic model of intention reading and pattern finding that allows to learn an open-ended, bidirectional construction grammar through communicative interactions (Section 7.2.5).

7.2.1 Large-Scale Computational Construction Grammar

I have developed a computational construction grammar with FCG and accompanying procedural semantic representation with IRL that together solve the CLEVR benchmark task on the symbolic level. Concretely, the constructions cover all questions of the dataset in both the comprehension and production direction. In other words, all questions can
be correctly mapped to their underlying meaning representation and vice versa. The procedural semantic representations allow to correctly compute the answer to all questions when executed on symbolic annotations of the CLEVR scenes. Both the constructions and the procedural semantic representations were designed on the basis of annotated data provided with the CLEVR dataset. Their respective processing mechanisms, i.e. constructional language processing and evaluation of semantic networks, are completely transparent and human-interpretable, support the integration of new constructions or primitive cognitive operators, and do not rely on annotated training data. Because of its 100% coverage and accuracy, parts of this system can be confidently re-used in subsequent chapters of this dissertation as scaffolding or as the gold-standard. This computational construction grammar corroborates the theoretical underpinnings of the field of construction grammar, e.g. by demonstrating the lexicon-grammar continuum and the tight integration of morphosyntax and semantics, and contributes to the scaling of computational construction grammars. In particular, this grammar is one of the first grammars that operates on this scale, covering more than one million utterances in both directions of processing. It thereby demonstrates the capabilities of Fluid Construction Grammar for operationalising large-scale constructionist approaches to language. An interactive web demonstration of the CLEVR grammar can be found at https://ehai.ai.vub.ac.be/demos/clevr-grammar.

7.2.2 Learning Grounded Concepts through Discrimination

I have presented an interactive learning approach, through the language game paradigm, which allows an autonomous agent to extract meaningful, symbolic concepts from continuous streams of sensorimotor data and thereby bridge the gap from low-level observations to higher-level reasoning and communication. The concept representation is inspired by prototype theory (Rosch, 1973) and the learning mechanisms extend earlier work within the language game paradigm (Wellens, 2012). Learning these concepts in terms of continuous data-streams requires the agent to simultaneously extract relevant features and determine their prototypical values. The notion of discrimination plays a central role in overcoming this learning problem. Through a series of communicative interactions modelling an object reference task, the agent was tasked with learning the various perceptual concepts that are present in the CLEVR dataset. This task was operationalised in two settings which differ in the sensorimotor data-stream provided to the agent, namely the simulated setting and the noisy setting. In both settings, the agent could successfully acquire the perceptual concepts that are present in the CLEVR dataset and use them near-perfectly in bidirectional communication. Dedicated experiments were set up to investigate various aspects of the presented approach, including its data-efficiency, generality, transparency and adaptivity. The experimental results have convincingly shown that the approach exhibits these desirable properties and thereby make it highly valuable for the domains of robotics and interactive task learning, where fast, data-efficient, and adaptive learning mechanisms grounded in the environment are crucial. Moreover, the approach advances
7.2. ACHIEVEMENTS

the research on the emergence and acquisition of conceptual systems within the language game paradigm.

### 7.2.3 Fully Explainable Grounded Language Processing System

I have demonstrated the integration of grounded, symbolic concepts in a higher-level reasoning task. Concretely, an agent equipped with the CLEVR grammar uses the acquired repertoire of concepts from the previous experiment to ground the lexical items occurring in the CLEVR questions in its sensorimotor data streams. This integration constitutes a language processing system that is fully explainable, ranging from the low-level perception and categorisation to the higher-level language processing and reasoning. This integration was evaluated through the CLEVR benchmark task. On the one hand, the evaluation has revealed that inaccuracies in the acquired concepts can propagate and accumulate through the semantic networks, and highlighted the crucial role of the feature extraction process prior to the concept learning methodology. On the other hand, the evaluation has demonstrated that the system will only produce answers when it can confidently infer them, and yielded results that are competitive with state-of-the-art approaches in the simulated setting.

### 7.2.4 Hybrid Procedural Semantics

I have presented hybrid procedural semantics as a methodology that elegantly and flexibly combines the strengths of symbolic operations on structured data with sub-symbolic operations on unstructured data through procedural semantic representations. Specifically, primitives operating on the sub-symbolic level are operationalised by small and modular neural networks that perform classification or semantic segmentation tasks directly on image data. These are combined with primitives operating on the symbolic level performing higher-level reasoning tasks on structured representations. The modular neural networks are highly specialised such that each network corresponds to one particular concept that is grounded in image data. Additionally, each neural module produces human-interpretable outputs in the form of symbolic labels (i.e. through classification) or image masks (i.e. through semantic segmentation). These two aspects greatly enhance the system’s overall transparency since the source of potential reasoning errors can be more easily traced back to one specific neural network and the output of these neural networks can be visually inspected. Furthermore, by separating each concept in distinct, yet modular, neural networks, the repertoire of available concepts can be easily expanded without needing to retrain or adjust existing components. Evaluation of hybrid procedural semantics on the CLEVR benchmark task resulted in a level of accuracy that is competitive with state-of-the-art approaches which, crucially, do not possess the aforementioned highly desirable properties. Given its dependency on trained neural modules, the hybrid procedural semantics approach is adequate for operationalising transparent and explainable processing
in intelligent systems that tackle a specific task.

### 7.2.5 Mechanistic Model of Intention Reading and Pattern Finding

For the most important achievement of this dissertation, I have developed a mechanistic model of the cognitive capacities of intention reading and pattern finding, as described by Tomasello (2003, 2009b). This mechanistic model allows an autonomous agent to incrementally and efficiently acquire an open-ended grammar, in the form of an inventory of constructions, that is well suited for solving a communicative task. Concretely, a learner agent is provided with a number of primitive cognitive operators, which serve as the building blocks of meaning, and human-interpretable learning operators that implement intention reading, pattern finding and entrenchment dynamics. These learning operators are operationalised using the agent’s meta-level architecture. Together with a tutor agent, the learner engages in a language game where they ask and answer questions to each other, called the elicitation game. When a game cannot be completed, the learner’s meta-level operators kick in. On the one hand, intention reading allows the agent to hypothesise about a possible meaning underlying an observed question. This is implemented in IRL through the composition of cognitive operators into a semantic network that allows to infer the interlocutor’s intention. The addition of a memory component that stores past experiences to IRL’s composer mechanism was crucial in order to overcome incorrect meaning hypotheses. On the other hand, pattern finding consists of a number of strategies for generalising over observed questions and reconstructed meanings, and for capturing a network of emergent grammatical categories. These strategies are based on work by Doumen et al. (forthcoming) and have been extended to incorporate intention reading. Specifically, the pattern finding strategies allow the agent to transition from holophrastic mappings between forms and meanings to item-based and lexical mappings, which provide insights into the compositional and non-compositional aspects of the language. The agent’s meta-level tries out several of these strategies in order to extend the agent’s inventory of constructions, such that it is better suited for future question answering and covers not only the observed questions, but also similar novel ones.

The major challenge in the acquisition of a construction grammar through communicative interactions is the search space faced by intention reading, as this consists of every possible meaning that could underlie an utterance in the given context. This search space is infinitely large and difficult to navigate. Moreover, only indirect supervision is available as the agent can only observe the context, the utterance and the interlocutor’s intention, but never the underlying meaning.

I have argued, and shown through the presented experiments, that it is the interplay of intention reading and pattern finding that allows to overcome this challenge. Specifically, intention reading provides meaning hypotheses, which are necessary for bootstrapping the pattern finding strategies. In turn, generalisations made through pattern finding allow
for a partial understanding of previously unobserved utterances. The partial meanings fix part of the search space faced by intention reading, thereby reducing the search problem. Combined with mechanisms that model the entrenchment of constructions, this allows the agent to overcome competing form-meaning hypotheses, keep constructions that can be used successfully in communication and discard unsuccessful constructions, and thus gradually shape an effective and efficient construction grammar that is adequate for solving the communicative task, namely asking and answering questions.

In sum, I have presented a number of transparent and human-interpretable learning mechanisms that together model the cognitive capacities of intention reading and pattern finding, and entrenchment dynamics. These allow the agent, which is only equipped with a number of primitive cognitive operators, to engage in the highly non-trivial processes of meaning reconstruction and schema abstraction in order to learn a communication system in the form of a construction grammar that solves the communicative task in the environment. Similar to Doumen et al. (forthcoming), these cognitive capacities have been implemented as truthfully as possible with respect to Tomasello (2003)'s work and the same outcomes as empirically observed have been obtained. Thereby, this experiment provides computational evidence for the cognitive plausibility of these theories. Also similar to Doumen et al. (forthcoming), the experiment demonstrates how the operationalisation of theoretical findings from the field of construction grammar result in a computational account of usage-based construction grammar learning. Finally, and most importantly, the experiment pushes forward the state of the art in the development of human-like communication systems for autonomous agents and paves the way for future intelligent agents that can acquire sophisticated communication systems with human-like properties through communicative interactions.

7.3 Limitations and Avenues for Future Research

In this section, I discuss a number of limitations of the achievements presented in Section 7.2, together with avenues for future research that come within reach by building further on my contributions and addressing these limitations. Concretely, I discuss (i) the integration of my achievements (Section 7.3.1), (ii) the evaluation of the agent’s representations and learning mechanisms from an end-user’s perspective (Section 7.3.2), (iii) the extension of methodologies for learning conceptual structures beyond visual concepts (Section 7.3.3), (iv) the transition from language acquisition to language emergence experiments (Section 7.3.4), (v) the problem of determining the symbolic/sub-symbolic boundary in hybrid systems (Section 7.3.5) and (vi) the scaling of the mechanistic model of intention reading and pattern finding, in terms of the language, modality and complexity of the input (Section 7.3.6).
7.3.1 Integration

One direction of future research consists in the integration of the methodologies for learning conceptual structures and for learning morpho-syntactic and semantic structures. Concretely, both the grounded concept learning methodology of Chapter 4 and the hybrid procedural semantics approach of Chapter 5 can be integrated with the methodology for learning construction grammars from Chapter 6. Integrating these methodologies, particularly Chapters 4 and 6, would allow to investigate a number of questions concerning the interplay of learning concepts and learning grammars. For example, should the visual concepts be acquired prior to the acquisition of grammatical structures or should these learning processes overlap? Should there be feedback loops between the learning mechanisms for these two types of linguistic structures? How does the conceptualisation of the environment influence the learning of grammatical structures? Should semantic categories that group together related concepts be learned separately, or do they emerge through language use during grammar learning? Using the CLEVR dataset as an example, integrating these methodologies would constitute a language acquisition experiment where the agent learns both (i) an open-ended repertoire of human-interpretable visual concepts from continuous, sensorimotor data, and (ii) an effective and efficient open-ended, bidirectional grammar that can be used to ask and answer questions about the objects in the environment. The results of these experiments may further contribute to a better understanding of the mechanisms underlying language acquisition through situated, communicative interactions and their operationalisation in autonomous agents.

7.3.2 Evaluating Explainability

As part of the primary objective of this dissertation, I focus on representations and learning mechanisms that are designed to be transparent and human-interpretable. This explainability is considered from a technical perspective. Specifically, throughout my contributions, I have made use of symbolic feature structures, sets of predicates, networks, and numerical values representing normal distributions, scores, probabilities, etc. The symbolic structures make use of symbols that are meaningful from a human observant perspective and the numerical structures can be straightforwardly interpreted in terms of the task or the environment. This in contrast to, for example, neural networks where additional procedures need to be ran in order to visualise and interpret the weights of trained networks. What was not considered in this dissertation is the viewpoint of Explainable Agency (Langley et al., 2017), where an agent actively explains its own decisions and reasoning processes, or an end-user perspective, e.g. a person interacting with an autonomous agent that acquires a grammar through intention reading and pattern finding. In case of the former, I argued that the use of symbolic structures facilitates this explanation process. The latter, namely whether end-users find the agent’s representations useful and are able to interpret what the agent is learning, would require additional evaluation, e.g. through questionnaires and user studies. This could be pursued in future research in collaboration
7.3. LIMITATIONS AND AVENUES FOR FUTURE RESEARCH

with experts in the field of human-robot interaction or digital media and society.

7.3.3 Beyond Visual Concepts

Both methodologies for learning conceptual structures (Chapter 4 and Chapter 5) focus on visual concepts. To expand the communicative capabilities of autonomous agents, future research could build further on these methodologies in order to go beyond visual concepts.

In terms of Chapter 4, the same methodology could be explored for learning concepts that are based on function, audio or timing rather than visual perception. These kinds of concepts are often learned prelinguistically and require operationalising the presented methodology in a richer environment where these other modalities are available, e.g. through virtual reality where all aspects of the environment can be exactly parametrised.

The hybrid procedural semantics methodology of Chapter 5 can be extended to incorporate multiple sources of information. In one such extension, developed by Verheyen et al. (submitted), the hybrid procedural semantics methodology has been extended from a Visual Question Answering task (Antol et al., 2015) to a Visual Dialogue task (Das et al., 2017a). In this task, the system has to keep track of the information that is incrementally conveyed during a dialogue. This information is stored in a symbolic data structure, called the conversion memory, that explicitly and incrementally represents the information that is expressed in subsequent turns of a dialogue. The strength of hybrid procedural semantics is that it allows to elegantly and flexibly reason over perceptual observations and the conversation memory simultaneously. A similar extension could focus on combining information from images with large-scale knowledge graphs. Such an extension exploits the strengths of hybrid procedural semantics for tasks that require reasoning over perceptual observations and ontological knowledge. The results of both of these experiments may contribute to the growing body of work in artificial intelligence that tackles tasks through a combination of sub-symbolic processing over unstructured data and symbolic reasoning over structured data (see e.g. Yi et al. (2018); Manhaeve et al. (2018, 2021); Dumančić et al. (2019); Mandi et al. (2020); Badreddine et al. (2022); van Krieken et al. (2022).

7.3.4 From Language Acquisition to Language Emergence

In Chapters 4 and 6, I have set up language game experiments that focus on the acquisition of visual concepts and construction grammars, respectively. These experiments were set up in a tutor-learner scenario, where a tutor has an established communication system and a learner is tasked with learning linguistic structures that allow it to communicate successfully with the tutor. Such a setting allows to investigate and operationalise the learning mechanisms that allow the agent to acquire a specific linguistic phenomenon. With these learning mechanisms in place, future work can address the emergence of visual concepts and construction grammars. Emergence experiments study how linguistic
phenomena can originate, spread, evolve and become conventionalised in a population of autonomous agents where none of these agents has an established communication system from the start (see also Section 2.2.2). In terms of the concept learning methodology of Chapter 4, a population setting would give the agents the freedom to select relevant combinations of feature channels for each concept. They are less bound to the concepts known by the tutor in a tutor-learner scenario. As a consequence, they might come up with feature combinations that do not necessarily correspond to visual concepts that are used in English, such as colours and shapes. This is not a problem in itself, as long as the concepts the agents come up with are well suited for the communicative task in the environment. Extending the construction grammar learning experiment of Chapter 6 to a population setting introduces a number of questions and challenges, most notably about how morpho-syntactic and semantic structures originate and evolve. These questions can be investigated through the language game experimental paradigm, and by taking inspiration from usage-based models of language (e.g. Tomasello (2003)), grammaticalisation (e.g. Hopper and Traugott (2003)) or other methodologies and processes from linguistics. The results of all of these experiments may contribute to a better understanding of the mechanisms through which natural languages can be acquired, emerge, and evolve through communicative interactions, and how human-like communication systems can be operationalised in autonomous agents.

7.3.5 Symbolic/Sub-Symbolic Boundary

The hybrid procedural semantics methodology presented in Chapter 5 is one of a growing number of systems that combines sub-symbolic processing over unstructured data (here image-processing neural networks) with symbolic processing over structured data (here higher-level reasoning) (see also Section 5.2.2). These approaches are aimed at providing more explainable systems that can perform symbolic reasoning while integrating with and maintaining high performance on perceptual tasks. The challenging aspect in developing these hybrid systems is where to draw the boundary between the two levels of processing. The view that was presented in Chapter 5 argues for using sub-symbolic processing for low-level perception tasks, while using symbolic processing for all higher-level reasoning. The sub-symbolic level thus abstract away over the perceptual level and provides meaningful symbols, which the symbolic level can use for reasoning. However, a limitation of the operationalisation presented in Chapter 5 is that it does not completely reflect this view. For instance, a separate neural module COUNT is trained for determining the arity of a set of objects, with the EXIST and UNIQUE modules rely on that module, while the GREATER-THAN module operates on the symbolic level. However, it is somewhat counter-intuitive to operationalise a counting operation on the sub-symbolic level, especially considering the fact that all objects have already been observed by another module, namely GET-CONTEXT. An important avenue of future research thus consists in shifting the symbolic/sub-symbolic boundary to better reflect the ideas presented in Chapter 5. Shifting this boundary requires
taking into account the trade-off between performance and explainability, especially on the sub-symbolic level. Concretely, the use of sub-symbolic processing for a larger portion of the task might increase the overall performance, as neural networks are generally able to achieve high levels of accuracy on perceptual tasks, but the overall explainability of the system is reduced. In contrast, using sub-symbolic processing for a smaller portion of the task shifts more responsibility to the symbolic reasoning component, and thereby increases the overall explainability. In general, inconsistencies in the reasoning process should be avoided at all cost. These can occur when results of sub-symbolic procedures contradict each other on the symbolic level, regardless of whether these results are correct or not.

### 7.3.6 Scaling Intention Reading and Pattern Finding

Many exciting experiments that build further on the methodology for learning construction grammars (Chapter 6) can be pursued. In particular, I discuss three avenues for future research. Broadly speaking, each of these avenues addresses the scaling of the presented methodology, either in terms of the language of the input, the modality of the input or the complexity of the input.

A first avenue of future research concerns the language of the input. Specifically, given that the presented learning mechanisms have been developed and tested using the English questions from the CLEVR dataset, they rely on the strict word order that is used in English in order to identify differences and similarities across constructions. This poses an issue for applying the exact same learning mechanisms to other languages with less strict word order or languages where other markers, such as case, number or gender, are used to indicate which words in an utterance belong together. In future work, pattern finding mechanisms that also take these other markers into account can be developed in order to increase the applicability of the mechanistic model of intention reading and pattern finding to different languages.

A second avenue of future research concerns the modality of the observed utterances. Specifically, the form side of constructions is represented using a textual representation, in particular sets of predicates that capture tokens from the utterance and adjacency relations between them. However, the problem that children face in terms of pattern finding is far more difficult, as they observe utterances in terms of speech signals. In future work, the mechanistic model of intention reading and pattern finding can be extended to include speech signals. This will require algorithms that can identify and extract differences and similarities across those signals, and make compositional generalisations over them. The challenge here is that two speech signals are never exactly the same, and thus adequate similarity metrics are necessary to compute these generalisations. This avenue of future research can be pursued in collaboration with experts in the field of signal processing. Moreover, future research in this direction contributes to a growing body of work on multi-
modal construction grammar (see e.g. Steen and Turner (2013); Zima and Bergs (2017); Hoffmann (2021)).

A third avenue of future research consists in scaling the complexity of the inputs. While already operating on a larger scale, relying on fewer scaffolds and considering inputs of greater complexity compared to prior work (see Section 6.2.2), the presented methodology could not be applied to the entire CLEVR dataset. Scaling the complexity of the input can be achieved by presenting utterances with more complex morpho-syntactic structures on the one hand, and providing the agent with a larger inventory of primitive cognitive operators on the other hand. The latter is used to build more complex semantic structures. The introduction of more complex inputs requires technical advances both in terms of intention reading and pattern finding. In terms of intention reading, the chunking of semantic networks (see Section 2.4.6) would gain a more important role. Through chunking, the agent can learn conventionalised ways of construing semantics and use these in order to reduce the search space faced by intention reading. Chunking strategies, i.e. choosing which semantic (sub)network to conventionalise, together with entrenchment dynamics over chunks would need to be developed. In terms of pattern finding, more advanced strategies that support the acquisition of modular and recursive constructions are necessary. Specifically, modular constructions are constructions in which slots can be filled by other constructions which have slots themselves. Consequently, a recursive construction is construction where the open slots can be filled by the construction itself. Moving towards modular and recursive constructions requires lifting the restriction that is currently applied to pattern finding, namely that generalisation can only occur when a minimal difference between form and meaning (i.e. a difference of a single predicate) is found. In fact, this restriction could already be lifted in experiments for construction grammar learning from annotated corpora (Doumen et al., forthcoming), where both form and meaning are provided as input and thus intention reading is not necessary. There, it has allowed the agent to make more generalisations with fewer observations and to acquire modular and recursive constructions. In terms of the experiment presented in Chapter 6, the question would be how these more powerful generalisation capabilities cope with possibly incorrect meaning hypotheses that intention reading generates. The aforementioned mechanisms, namely chunking and modular constructions, are undoubtedly related and further experiments would also shed light on their interaction. Finally, when scaling to more complex morpho-syntactic and semantic structures, another aspect that is crucial is the ability to compute generalisations over them on the technical level. Concretely, both form and meaning are represented as unordered sets of predicates that are declaratively combined by sharing variables (see Section 2.3.4). The algorithm for computing differences and similarities is currently restricted to certain kinds of structures, e.g. sets of predicates where the variable links do not introduce cycles. However, cycles do occur in more elaborate semantic networks, e.g. in the CLEVR dataset. A more general approach for generalising over semantic networks would be to operationalise the process of anti-
7.4. FINAL REMARKS

unification, which allows to find the least general generalisation (LGG) over two symbolic expressions. While most approaches focus on anti-unification over ordered symbolic expressions, here, anti-unification over unordered sets of predicates is required, e.g. as in Yernaux and Vanhoof (2019, 2022). In terms of pattern finding, the LGG corresponds to the similarities across two semantic networks, while the symbols or predicates that the anti-unification process generalises over can be used to identify the differences between the two semantic networks.

In sum, the avenues for future research that are described above include both methodological innovations and technical advances, and may further extend the powerful methodology for learning sophisticated communication systems with human-like properties through communicative interactions to include multiple languages, multiple modalities and the acquisition of more complex morpho-syntactic and semantic structures.

7.4 Final Remarks

In my dissertation, I have introduced novel representations and learning mechanisms that allow autonomous agents to acquire linguistic structures on the conceptual, morpho-syntactic and semantic level. The agents incorporate these linguistic structures in communication systems that exhibit highly desirable properties also found in human languages, such as robustness, flexibility, adaptivity, learning efficiency and expressiveness. Moreover, from a computational point of view, the introduced representations and learning mechanisms are transparent and explainable in human-interpretable terms, thereby allowing to more easily validate their internal correctness and consistency, and elicit trust. I am confident that the proposed representations and learning mechanisms as developed in this dissertation can serve three main purposes. First, the developments in terms of (computational) construction grammar, such as contributing to the scaling of computational construction grammars and the methodology for usage-based learning of construction grammars, contribute to and accelerate the wide-spread use of constructionist approaches to language in many sub-fields of linguistics, such as historical linguistics, language acquisition, evolution and change, language learning and teaching, and psycholinguistics, and corroborate many of the theoretical findings of the field. Second, my contributions can lead to more advanced agent-based experiments on the origins, emergence and evolution of natural languages in the field of evolutionary linguistics, especially concerning grammatical structures. In turn, these experiments allow to gain more insights into human languages and cognition, and contribute to the hypothesis that linguistic structures are emergent through gradual evolutionary processes taking place during communicative interactions. Finally, this dissertation contributes to the development of future intelligent systems where autonomous agents interact among themselves or with human interlocutors through natural language in order to solve a particular task, such as visual question answering systems, conversational agents, personal assistants, human-robot interaction
systems and intelligent tutoring systems. In these systems, truly intelligent communicative behavior can only be obtained through precise mechanistic models of the mechanisms underlying the acquisition and evolution of human languages, which is exactly what the representations and learning mechanisms presented in this dissertation contribute to. By building further on these contributions, the next wave of intelligent systems will have robust, bidirectional language processing capabilities that are tailored to the environment, support an open-ended set of tasks, remain ever-adaptive to a changing environment, and can explain their own reasoning processes.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


Appendix A

Supplementary Materials for Chapter 4

Figure A.1: All concepts acquired in one experimental run in the simulated environment.
Figure A.1: All concepts acquired in one experimental run in the simulated environment.
Figure A.1: All concepts acquired in one experimental run in the simulated environment.

Figure A.2: All concepts acquired in one experimental run in the extracted environment.
Figure A.2: All concepts acquired in one experimental run in the extracted environment.
Figure A.2: All concepts acquired in one experimental run in the extracted environment.

Figure A.3: Shape-related concept acquired in the simulated environment during the generalisation experiment.
Figure A.4: Shape-related concept acquired in the noisy environment during the generalisation experiment.
Appendix B

Supplementary Materials for Chapter 5

Table B.1: Evaluation results of the neural modules on a held-out test set on the CoGenT dataset (condition A).

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Loss (BCE)</th>
<th>Accuracy (%)</th>
<th>Neural Module</th>
<th>Loss (NLL)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-context</td>
<td>0.008</td>
<td>99.36</td>
<td>count</td>
<td>0.028</td>
<td>99.47</td>
</tr>
<tr>
<td>find[blue]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[blue]</td>
<td>0.004</td>
<td>99.97</td>
</tr>
<tr>
<td>find[brown]</td>
<td>0.001</td>
<td>99.91</td>
<td>query[brown]</td>
<td>0.004</td>
<td>99.97</td>
</tr>
<tr>
<td>find[cube]</td>
<td>0.004</td>
<td>99.78</td>
<td>query[cube]</td>
<td>0.011</td>
<td>99.86</td>
</tr>
<tr>
<td>find[cyan]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[cyan]</td>
<td>0.005</td>
<td>99.95</td>
</tr>
<tr>
<td>find[cylinder]</td>
<td>0.003</td>
<td>99.80</td>
<td>query[cylinder]</td>
<td>0.007</td>
<td>99.90</td>
</tr>
<tr>
<td>find[gray]</td>
<td>0.002</td>
<td>99.91</td>
<td>query[gray]</td>
<td>0.005</td>
<td>99.96</td>
</tr>
<tr>
<td>find[green]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[green]</td>
<td>0.006</td>
<td>99.94</td>
</tr>
<tr>
<td>find[large]</td>
<td>0.005</td>
<td>99.59</td>
<td>query[large]</td>
<td>0.011</td>
<td>99.86</td>
</tr>
<tr>
<td>find[metal]</td>
<td>0.004</td>
<td>99.70</td>
<td>query[metal]</td>
<td>0.007</td>
<td>99.89</td>
</tr>
<tr>
<td>find[purple]</td>
<td>0.001</td>
<td>99.93</td>
<td>query[purple]</td>
<td>0.004</td>
<td>99.96</td>
</tr>
<tr>
<td>find[red]</td>
<td>0.001</td>
<td>99.93</td>
<td>query[red]</td>
<td>0.005</td>
<td>99.94</td>
</tr>
<tr>
<td>find[rubber]</td>
<td>0.004</td>
<td>99.72</td>
<td>query[rubber]</td>
<td>0.009</td>
<td>99.89</td>
</tr>
<tr>
<td>find[sphere]</td>
<td>0.002</td>
<td>99.85</td>
<td>query[sphere]</td>
<td>0.009</td>
<td>99.86</td>
</tr>
<tr>
<td>find[yellow]</td>
<td>0.001</td>
<td>99.91</td>
<td>query[yellow]</td>
<td>0.005</td>
<td>99.95</td>
</tr>
<tr>
<td>relate[behind]</td>
<td>0.004</td>
<td>99.73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[front]</td>
<td>0.005</td>
<td>99.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[left]</td>
<td>0.003</td>
<td>99.73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[right]</td>
<td>0.003</td>
<td>99.74</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table B.2: Evaluation results of the neural modules on a held-out test set on the CoGenT dataset (condition B).

<table>
<thead>
<tr>
<th>Neural Module</th>
<th>Loss (BCE)</th>
<th>Accuracy (%)</th>
<th>Neural Module</th>
<th>Loss (NLL)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-context</td>
<td>0.009</td>
<td>99.35</td>
<td>count</td>
<td>0.024</td>
<td>99.51</td>
</tr>
<tr>
<td>find[blue]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[blue]</td>
<td>0.006</td>
<td>99.94</td>
</tr>
<tr>
<td>find[brown]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[brown]</td>
<td>0.002</td>
<td>99.99</td>
</tr>
<tr>
<td>find[cube]</td>
<td>1.355</td>
<td>81.58</td>
<td>query[cube]</td>
<td>5.417</td>
<td>33.19</td>
</tr>
<tr>
<td>find[cyan]</td>
<td>0.001</td>
<td>99.90</td>
<td>query[cyan]</td>
<td>0.003</td>
<td>99.97</td>
</tr>
<tr>
<td>find[cylinder]</td>
<td>0.872</td>
<td>81.52</td>
<td>query[cylinder]</td>
<td>5.038</td>
<td>35.77</td>
</tr>
<tr>
<td>find[gray]</td>
<td>0.002</td>
<td>99.91</td>
<td>query[gray]</td>
<td>0.005</td>
<td>99.95</td>
</tr>
<tr>
<td>find[green]</td>
<td>0.002</td>
<td>99.90</td>
<td>query[green]</td>
<td>0.004</td>
<td>99.96</td>
</tr>
<tr>
<td>find[large]</td>
<td>0.006</td>
<td>99.57</td>
<td>query[large]</td>
<td>0.012</td>
<td>99.87</td>
</tr>
<tr>
<td>find[metal]</td>
<td>0.004</td>
<td>99.69</td>
<td>query[metal]</td>
<td>0.008</td>
<td>99.89</td>
</tr>
<tr>
<td>find[purple]</td>
<td>0.002</td>
<td>99.90</td>
<td>query[purple]</td>
<td>0.005</td>
<td>99.96</td>
</tr>
<tr>
<td>find[red]</td>
<td>0.002</td>
<td>99.90</td>
<td>query[red]</td>
<td>0.004</td>
<td>99.95</td>
</tr>
<tr>
<td>find[rubber]</td>
<td>0.004</td>
<td>99.70</td>
<td>query[rubber]</td>
<td>0.009</td>
<td>99.88</td>
</tr>
<tr>
<td>find[small]</td>
<td>0.003</td>
<td>99.84</td>
<td>query[small]</td>
<td>0.009</td>
<td>99.89</td>
</tr>
<tr>
<td>find[sphere]</td>
<td>0.002</td>
<td>99.86</td>
<td>query[sphere]</td>
<td>0.006</td>
<td>99.90</td>
</tr>
<tr>
<td>find[yellow]</td>
<td>0.001</td>
<td>99.92</td>
<td>query[yellow]</td>
<td>0.003</td>
<td>99.98</td>
</tr>
<tr>
<td>relate[behind]</td>
<td>0.004</td>
<td>99.73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[front]</td>
<td>0.005</td>
<td>99.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[left]</td>
<td>0.003</td>
<td>99.73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relate[right]</td>
<td>0.003</td>
<td>99.73</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table B.3: Performance of the hybrid procedural semantics approach on both condition A and B of the CLEVR CoGenT dataset \textit{before finetuning}. The question answering accuracy is split per question type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Exist</th>
<th>Count</th>
<th>Compare Integer</th>
<th>Query</th>
<th>Compare Attribute</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>equal</td>
<td>less</td>
<td>more</td>
<td>size</td>
</tr>
<tr>
<td>A</td>
<td>99.6</td>
<td>98.5</td>
<td>99.6</td>
<td>99.6</td>
<td>99.7</td>
<td>99.8</td>
</tr>
<tr>
<td>B</td>
<td>84.3</td>
<td>70.5</td>
<td>73.9</td>
<td>78.8</td>
<td>81.6</td>
<td>79.4</td>
</tr>
</tbody>
</table>

Table B.4: Performance of the hybrid procedural semantics approach on both condition A and B of the CLEVR CoGenT dataset \textit{after finetuning}. The question answering accuracy is split per question type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Exist</th>
<th>Count</th>
<th>Compare Integer</th>
<th>Query</th>
<th>Compare Attribute</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>equal</td>
<td>less</td>
<td>more</td>
<td>size</td>
</tr>
<tr>
<td>A</td>
<td>83.8</td>
<td>68.6</td>
<td>73.7</td>
<td>80.4</td>
<td>84.6</td>
<td>80.8</td>
</tr>
<tr>
<td>B</td>
<td>96.2</td>
<td>88.1</td>
<td>94.1</td>
<td>95.9</td>
<td>96.4</td>
<td>97.3</td>
</tr>
</tbody>
</table>
Appendix C

List of Publications

The papers that I have published during my PhD project (2017-2022) are listed below. A complete list of my talks, posters and research activities is accessible via https://ai.vub.ac.be/members/jens-nevens.


