

Vrije Universiteit Brussel
Faculteit Wetenschappen en
Bio-ingenieurswetenschappen
Departement Computerwetenschappen

Towards an agent-based tutoring system for Spanish verb conjugation

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

Katrien BEULS

Promotor: Prof. dr. Luc Steels Brussel, november 2013

Abstract

Mobile devices such as tablets, smart phones or e-readers have become omnipresent in our daily lives for reading the news, analyzing our latest workout data, checking a cooking recipe, but also to learn the basics of a new language. Language learning applications are becoming an interesting alternative to classroom education because they allow learners to practice anywhere and at any time, often even for free. Yet, the quality of tutoring they offer can usually not be compared with a human tutoring scenario, in which the tutor analyses the learner's language skills and adapts the curriculum accordingly, but they lie closer to programmed instruction systems where every user proceeds through the same sequence of learning steps. To build an artificial tutor that can process the learner's input and react to it in an individualised way three key ingredients are needed: (i) a fully operational grammar of the target language that can parse and produce any learner utterance, even when it is ungrammatical; (ii) a predictive student model that can simulate the real student and is aligned to match the student's linguistic skills after every interaction with the system; (iii) a set of tutoring strategies that is used to guide the student through the learning programme. This dissertation presents operational solutions for (i) and partially (ii) for the domain of Spanish verb conjugation and an agent-based tutoring design accompanied by preliminary experiments for (iii). The solutions include a competent language agent that has a complete grammar to conjugate any Spanish verb form and repair learner errors and a runnable student agent that can acquire the target grammar incrementally through situated learning in discriminative contexts. The successful evaluation of these components has led to the constructive design of a tutor agent that incorporates both language and student agents and engages in interactions with the real student, whom he assists on his learning path with carefully tailored tutoring strategies.

Samenvatting

Mobiele apparaten zoals tablets, smart phones en e-readers zijn momenteel niet meer weg te denken uit ons dagelijkse bestaan. We gebruiken ze om de krant te lezen, een nieuw recept op te zoeken, sportieve prestaties te analyseren maar ook steeds vaker om een nieuwe taal aan te leren. Leerapplicaties worden steeds meer gezien als een interessant alternatief voor klassikaal taalonderwijs omdat zij leerders toestaan om nieuwe kennis te vergaren waar en wanneer zij hier behoefte aan hebben, en dit in vele gevallen gratis of tegen een lage prijs. De kwaliteit van de lessen die zij aanbieden kan meestal echter nog niet vergeleken worden met een klassituatie waarin de leraar de taalvaardigheden van zijn leerlingen inschat en de lessen daarop afstemt, maar leunt dichter aan bij de geprogrammeerde instructiesystemen waarin elke gebruiker dezelfde leerniveaus en -hindernissen doorloopt. Om een kunstmatige leraar te maken die talige input van een student kan verwerken en daarop op een aangepaste manier kan reageren zijn er drie bouwstenen nodig: (i) een volledig operationele grammatica van de doeltaal waarmee eender welke taaluiting kan geanalyseerd en geproduceerd worden, inclusief niet-grammaticale uitingen; (ii) een voorspellend studentenmodel dat een echte student simuleert en steeds afgestemd blijft op de taalvaardigheden van deze student; en (iii) een reeks tutoring strategieën die gebruikt worden om een student persoonlijk te begeleiden doorheen de leerstof. Deze thesis presenteert werkende oplossingen voor componenten (i) en (ii) voor het domein van Spaanse werkwoordsvervoegingen en een agent-gebaseerd tutoring design voor component (iii). De oplossingen bevatten een competente taalagent die over een volledige grammatica beschikt om enerzijds eender welke Spaanse werkwoordsvorm te vervoegen en anderzijds taalfouten op te sporen en te verbeteren, en een actieve studentenagent die de doelgrammatica kan verwerven door het spelen van gesitueerde taalspellen. De succesvolle evaluatie van deze componenten heeft geleid tot een eerste design van een tutor agent die kan communiceren met een student die hij op zijn leerpad bijstaat met zorgvuldig uitgezochte tutoring strategieën.

Acknowledgements

What began as an offsite meeting in the French alps in January 2009 has lead to the successful completion of a PhD thesis almost five years later. It was the peak days of the ALEAR project and the team was vibrant with scientific curiosity and energy to build new cognitive systems and tackle novel linguistic challenges. I immediately knew that I wanted to become part of this team. Already since the beginning, I have always experienced the freedom to pursue my own research interests and take new initiatives, thanks to stimulating research climate that my supervisor, Luc Steels, has created. Without his constant inspiration and rebellion to think out of the box I would not have become the researcher who I am today. Therefore, Luc, thank you very much for all the opportunities that you have given me to actively participate in interdisciplinary science and to meet outstanding researchers in different fields.

I also want to thank the other members of my PhD jury, Prof. Dr. Viviane Jonckers, Prof. Dr. Ann Nowé, Prof. Dr. Alex Housen, Prof. Dr. Piet Desmet and Prof. Dr. Emmanuel Keuleers for agreeing to read, comment on and judge this dissertation. Their comments and questions have significantly improved the quality of this work and put it in the right perspective. I also want to thank my Brussels colleague Johan for meticulously proofreading large parts of earlier versions of this dissertation and providing essential feedback and my Barcelona colleague Emília for helping me to structure my thoughts in stimulating discussions at our whiteboard.

All of my former and current colleagues, spread across three different labs in Brussels, Paris and Barcelona, also deserve a big thank you for their input on various moments during the last four years. In their order of appearance since the first offsite meeting in Morzine: Martin, who has taught me to strive for aesthetic beauty in the design of code, slides or interfaces; Pieter, literally the "rock" in my research existence who was always ready for inspiring office or lunch discussions about terminological or technical details of our work, thanks for your unconditional support; Remi, in many respects the papa Smurf of the team, giving good advice and at the same time using his wisdom to creatively advance the current state of the art in Fluid Construction Grammar (FCG) and Evolutionary Linguistics; Joachim, a free thinker who was always ready to hunt down some persistent bugs in FCG; Joris, who helped me with my IWT proposal and

introduced me to some of his favourite places in Brussels, which I still frequent; Vanessa and Katya, for a long time the only other women on the team, thanks for breaking the ice for me; Simon, who has impressed me multiple times with his determination to shake some fundamentals of traditional semantics and bridge FCG and IRL in his work; Michael, the technological centipede of the team who has always struck me with his unstoppable work energy; Nancy, always open for good discussions or entertaining story-telling during my time in Paris, Kevin, whom I actually met more than one year before I started my PhD but later turned into my office mate for two years, with his healthy curiosity to always question current design decisions and experimental conditions; Bart, who brought fresh energy into the Brussels lab when it was most needed; Emília, my Catalan best friend who has intensively followed the genesis of the writing of this dissertation during my two-month stay in Barcelona earlier this year and for whom Spanish verb conjugation still remains largely a mystery, *gràcies per tot!*; Johan, our lab musician, champagne lover and excellent proofreader who can always shake your mind setting with his witty remarks, thanks for reading this dissertation from the first to the last letter and your interest in FCG, your input is much appreciated; Miquel, the lab's rugby guru who joined our team recently but who has already proven his team-building and coaching capacities; Thank you all for all the incredible memories that we have had together! I further want to thank the long-lasting support, whether technical, administrative or moral, of Frederik, Lara, Carl and Ellen. Matthias also deserves a big thank you for this beautiful LaTeX thesis template that I could borrow and his expert LaTeX help on Skype at some critical moments.

I want to thank my parents, Bart and Renée, for always supporting me in everything that I do and giving me the opportunities to study abroad on different occasions during my pre-doctoral studies; they have broadened my world. Also my four grandparents, thank you for genuinely showing your interest in my research and challenging me with very relevant questions. Thank you for being my biggest fans! My sister Barbara and brother Arnold, thank you for understanding the choices that I have made, I am sure that you will also find your own ways in life. My friends, your unconditional support has been invaluable to me especially in the final phases of writing this dissertation. You know who you are but I especially want to thank: the Letteren-gang, Liesbeth, Robbert, Karen, Kris, Marie, Jeff, Mieke and Tom; my secondary school Lyceum-friends, Marlies, Céline, Hanne, Katrien, Lore, Cynthia and Birgit and my Rotaract friends, Karel, Steven en Lara, Natacha en Wim, Katrin en Pieterjan, and all the Rotaract Leuven members.

Finally, my warmest appreciation goes to the person who has been with me through everything that happened the last four years, who has supported me in all the decisions that I made and who understands me without words: Thank you, Alex, for everything!

The research reported in this dissertation has been financially supported by a doctoral grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

Contents

1	Introduction	1
1.1	Objectives	2
1.1.1	Accurate language processing	3
1.1.2	Agent-based modelling	4
1.1.3	Meta-level strategies	5
1.2	Situating this work	6
1.2.1	Grammar formalisms	6
1.2.2	Error correction	7
1.2.3	Adaptivity	8
1.3	Towards an artificial language tutor	8
1.3.1	Modeling a competent speaker	9
1.3.2	A predictive student model	11
1.3.3	Tutoring strategies	12
1.4	Structure of the dissertation	14
I	Background	17
2	Artificial Intelligence in Education	19
2.1	A brief history of Intelligent Tutoring Systems	20
2.1.1	From frame-based approaches to ITSs	21
2.1.2	Language learning tutors	23
2.1.3	A third generation of ITS research	24
2.2	The general ITS architecture	26
2.2.1	Expert knowledge	28
2.2.2	Student knowledge	28
2.2.3	Tutoring knowledge	33
2.2.4	Constructing an ITS	34
2.3	Examples of language tutors	35
2.3.1	Academic language tutors	36
2.3.2	Commercial language tutors	39
2.4	Conclusion	44

II	Operationalization	47
3	Fluid Construction Grammar	49
3.1	The construction inventory	51
3.1.1	Constructions	51
3.1.2	Organizing constructions	58
3.1.3	Templates	61
3.2	The grammar engine	62
3.2.1	Match and merge	62
3.2.2	Searching constructions	64
3.3	Conclusion	65
4	Spanish verbs in FCG	69
4.1	A complex grammar	70
4.1.1	Formal complexity	71
4.1.2	Semantic complexity	73
4.2	A Spanish verb construction inventory	74
4.2.1	Lexical constructions	75
4.2.2	Phrasal constructions	78
4.2.3	Morphological constructions	79
4.2.4	Phonological constructions	80
4.3	A grammar engine for Spanish	82
4.3.1	Conjugating verbs with the FCG interpreter	82
4.3.2	FCG extensions	87
4.4	Related approaches in computational morphology	91
4.4.1	Finite-state morphology	91
4.4.2	Construction Grammar morphology	92
4.5	Conclusion	93
5	Flexibility strategies	95
5.1	Computational reflection	96
5.2	Meta-level flexibility strategies	98
5.2.1	Diagnostics	100
5.2.2	Repairs	102
5.2.3	Flexibility strategies interacting with language processing	103
5.3	FCG for self- and other-repair	106
5.4	Conclusion	109
6	A language agent for Spanish verbs	111
6.1	Bootstrapping a Spanish verb grammar	112
6.1.1	A decision tree for automatic grammar creation	113
6.1.2	Conjugating Spanish verb forms	117
6.2	Diagnostics and repairs for verb form processing	120

6.2.1	Feature mismatch	121
6.2.2	Deviating form	123
6.2.3	Unknown stem/suffix	123
6.2.4	Remaining errors	124
6.3	Corpus-based error analysis	124
6.3.1	The learner corpus	125
6.3.2	Evaluation Results	126
6.4	Conclusion	129
7	Learning strategies	131
7.1	Problem-driven learning	132
7.2	Meta-level learning strategies	134
7.3	Attested learning strategies in SLA	137
7.3.1	Cognitive strategies	137
7.3.2	Meta-cognitive strategies	138
7.3.3	Social/affective strategies	139
7.4	Conclusion	141
8	A student agent for Spanish verbs	143
8.1	The language game	144
8.2	Learning problems under the looking glass	147
8.2.1	Learning the basic mappings	147
8.2.2	Stem changes	151
8.2.3	Verb classes	154
8.2.4	Results	157
8.3	Conclusion	166
III	Future outlook	167
9	Future outlook	169
9.1	Pedagogical vision	171
9.1.1	Learning environments	171
9.1.2	Motivation	175
9.2	Student model	179
9.2.1	The student agent	181
9.2.2	The student profile	183
9.2.3	Measures	187
9.3	Tutoring Strategies	188
9.3.1	Reusing the meta-level architecture	188
9.3.2	Tutoring strategies to create flow	191
9.3.3	Remaining tutor agent processes	195
9.4	The Spanish verb tutor	200
9.5	Conclusion	203

CONTENTS

IV	Conclusions	205
10	Conclusions	207
10.1	Achievements	208
10.1.1	Reconstructing Spanish verb conjugation	208
10.1.2	Learning strategies for Spanish verbs	210
10.2	Future work	211
10.3	Final remarks	213
	Bibliography	215
	Publications	231
	Glossary	235
	Index	239

Chapter 1

Introduction

We are all born as natural tutors, engaging in teaching sophisticated skills and acquired knowledge to our peers already from a very young age. Tutoring can be considered to be the main driver behind cultural evolution of cognitive capacities such as tool use and language. Learning these capacities requires more than imitation or reinforcement learning, but instead relies on higher-order skills that allow the learner to contextualize, make analogies and generalize over the learner input. Although we all dispose of this capacity, computerized tutors usually do not have the skill to adapt to their learners but instead fall back on pre-programmed instruction scripts. Commercial language tutors such as the popular *Rosetta Stone* belong to this type of "programmed-instruction" machines. For example, if a student revisits a particular lesson, he is exposed to the same materials as he has already encountered on previous interactions with the system.

Yet, already in the 1980s researchers have tried to put Artificial Intelligence (AI) techniques into the construction of Intelligent Tutoring Systems (ITSs). The problem of how to best represent knowledge within an intelligent system had been a major concern within the field of AI. The knowledge domain of ITSs ranges from well-defined domains such as mathematics and computer programming to ill-defined domains such as architecture and language. The subfield that focuses on ITSs for language learning is called Computer-Assisted Language Learning (CALL for short). Intelligent CALL concentrates on using Natural Language Processing (NLP) techniques to analyse learner language.

This dissertation advocates the use of *accurate language processing*, *agent-based modelling* and *meta-level strategies* with the aim of building a new kind of language tutoring system for adult second language (L2) learners. Because the construction of a completely new type of language tutor requires the work of more than one dissertation, I will concentrate here on the basic framework that is needed by this tutor and demonstrate its use for a case study of Spanish verb morphology. The ultimate idea is that every

individual student that interacts with this ICALL tutoring system would have his own personal language tutor agent at its disposal that addresses him at an appropriate level and selects new linguistic situations that are challenging enough to trigger learning. The basic framework that embraces such a personal language tutor consists of three main building blocks:

1. Because domain knowledge is a crucial prerequisite for any personal language tutor, whether human or artificial, it is necessary to have a **fully operational language agent** that can function as a competent language user.
2. A **predictive student model** in the form of an autonomous student agent with a structure that is identical to the language agent can be dynamically aligned to fit the real student's progress. The student agent learns "along" with the student that it models.
3. A language agent can take up the role of the tutor when he is endowed with a set of **tutoring strategies**, which make use of the information present in the student model as well as a more general student profile module.

Of course, the realisation of these three components requires the necessary technical skills and theoretical insights as it cuts across three fields of expertise: Intelligent CALL, linguistic theory and second language acquisition theory. The work presented in this dissertation focuses on its contributions in the field of Intelligent CALL (ICALL) and the perspective that is taken is always AI-based, rather than linguistics or pedagogy-oriented. To better delineate the current work, the following sections state its main objectives (Section 1.1) and situate it in the field of ICALL (Section 1.2). The basic components of the proposed tutoring architecture are described in further details in Section 1.3.

1.1 Objectives

The three main ideas that form the backbone of this dissertation are (1) *accurate language processing*, (2) *agent-based modelling* and (3) *meta-level strategies*. Each of these is explained in the current section. Although they can be applied to any language system, I have selected the Spanish verb phrase as the target language system. Within a language, a number of different subsystems are used instinctively by native speakers of the language. Examples of these kinds of subsystems include the tense system in English, the case grammar of Latin, the reflexive and reciprocal pronoun system of Spanish, the classifier system of Swahili, and so on. Spanish verb conjugation is another such subsystem. Acquiring Spanish verb conjugation does not only require an understanding of the use of temporal, aspectual and modal expressions but also implies an active knowl-

edge of the rich morpho-phonology of the Spanish language, with verb stem changes, assimilation processes and many irregularities.

1.1.1 Accurate language processing

The use of the term "accurate" in accurate language processing refers to the employment of a symbolic representation of linguistic knowledge and procedures that is error-free. An artificial language tutor that can produce and analyse linguistic expressions in the target language system, i.e. a subsystem in the language that is taught (e.g. prepositions, determiners, verb conjugation, case, etc.), needs to be fully reliable. Producing erroneous input or analysing learner language with an error margin can potentially lead to disastrous effects. Accurate language processing can be contrasted with approximate models of language that use probabilities to predict the next word during language production or analysis. Examples of approximate language processing models are N-gram language models and the use of recursive neural networks for parsing (also known as deep learning) (Socher, Lin, Ng, & Manning, 2011).

Similar to precision grammar, a formal grammar that aims at distinguishing ungrammatical from grammatical sentences, accurate language processing relates semantic representations to surface strings in parsing and production. Yet, while precision grammars such as the English Resource Grammar generally adhere to generative conceptions of linguistic theory, I adopt a cognitive-functional approach by choosing to employ Fluid Construction Grammar (FCG) for operationalising the artificial language tutor's target grammar (Steels, 2011, 2012a). Instead of distinguishing well-formed utterances from ungrammatical ones, FCG instead models "how speakers express their conceptualizations of the word through language (= *production*) and how listeners analyse utterances into meanings (= *parsing*)" (van Trijp, 2013). FCG uses non-typed feature structures to represent linguistic knowledge and relies on a two-step unification process (match/merge).

To be an accurate model of a speaker or listener that belongs to a particular linguistic community, FCG-based operationalizations are always *bidirectional*: they work both in parsing and production with the same grammar rules and the same grammar engine. Bidirectionality implies that when designing new grammar rules, the grammar engineer pays attention to both the triggering conditions as well as the effects they will cause in parsing and in production.

1.1.2 Agent-based modelling

An agent is an autonomous entity that pursues its own goals and learns according to the outcome of its own or other agents' actions. I employ the notion of agent to model a language user, be it a native speaker that masters a language completely or a language learner that still has gaps in its grammatical knowledge of the target language system. Such a linguistic agent engages in communicative interactions in which it calls upon its linguistic competence and performance models to reach certain communicative goals. In this dissertation, I refer to an interaction between two agents, be it human or artificial, as a *language game*. A language game is a routinised interaction script that contains all elements that two agents need for reaching a joint goal through linguistic communication (conceptualisation, production, parsing, interpretation).

Multi-agent systems have sometimes been considered as good candidates for building basic ITS infrastructures as they fulfil all the necessary requirements (Nkambou, Bourdeau, & V., 2010, p. 369): (i) they are made of different interconnected, complex components; (ii) they provide multiple, different and complementary services; (iii) each of their components is functionally autonomous; and (iv) they are equipped with specific knowledge structure and reasoning mechanisms. Agents are thus often decomposed by their function in the teaching and learning process, with for instance one evaluation agent, one modelling agent, one recording agent, one student agent, etc. (Sun, Joy, & Griffiths, 2007). Yet, the use of agents in this dissertation differs from such a distributed system in that every agent needs to be capable of taking the role of speaker or listener in a language game and is thus a fully operational model of a language user. Evaluation, recording and other tasks needed in a tutoring system are not considered as agent-specific tasks but rather seen as routine processes that do not require autonomous reasoning.

Moreover, the usefulness of agent technology in intelligent education systems is their contribution to make these systems adaptive, able to learn and dynamic by providing dynamic adaptation of domain knowledge and of behaviour of individual learners ((Razek, Frasson, & Kaltenbach, 2002), cited by (Sun et al., 2007)). Pedagogical agent-based systems are often used to monitor a particular project and enhance communication between members of a group (Beer & Whatley, 2002). Some researchers have designed agents for every course unit (Shang, Shi, & Chen, 2001) or assigned a new agent to a specific learning topic (Boicu et al., 2004). My approach differs from these existing uses of agents in tutoring systems as I rely on two agents: one that models a competent language user and another one that simulates a language learner. The latter can then function as an active student model that can run and try out solutions in parallel to predict a student's behaviour. Section 1.3 contains more details on the architecture of both agents.

1.1.3 Meta-level strategies

Finally, the last idea that this dissertation relies on is the use of a meta-level architecture for language processing. The idea of computational reflection, making computational algorithms more robust by allowing them to find solutions for unexpected processes, goes back to meta-level architectures for syntax checking and code debugging purposes in computer programming languages (Maes, 1988; Maes & Nardi, 1988). The basic idea is that there is an additional processing level on top of routine processing that monitors every step and signals problems when something unexpected has occurred. A number of repair strategies that have been manually encoded to solve a particular problem either by a very specific repair move or by more general analogical reasoning trigger so that the original processing can continue after a short repair intervention.

Every repair strategy comes with one or more diagnostics because problems that occur in different occasions might require the same repair strategy. But also, one problem that has been diagnosed at a certain point in the processing pipeline can potentially be repaired by more than one repair strategy. This dissertation presents strategies for three purposes:

1. **Flexibility strategies** are used to guarantee flexible language processing, both in parsing and production. This dissertation investigates their use for detecting and correcting form-related learner errors in parsing.
2. **Learning strategies** are employed by the student agent to simulate the learning process of the target language system through online learning in language games. Again, the implementation of learning strategies in this dissertation focuses on learning problems related to form rather than meaning.
3. **Tutoring strategies** do not repair linguistic knowledge directly but instead diagnose gaps in a student agent's knowledge and recurrent learning problems. They repair such problems by selecting appropriate exercises and providing constructive feedback. Due to time constraints, tutoring strategies have not been implemented in this dissertation.

Although the idea of meta-level strategies is not new, my contribution lies in their use inside the FCG grammar engine. Earlier experiments on language emergence and evolution by team members relied on their availability after an agent had completed a process such as conceptualization, production, parsing, etc. or after a complete turn was over in the turn-taking process between speaker and listener (Beuls, van Trijp, & Wellens, 2012). Through an extension of the diagnostics and repairs framework I made it possible to diagnose problems after every search node in linguistic parsing or production, which allows for a faster diagnosis and the possibility to repair errors by relaxing constraints

or replacing certain features. Chapters 5 and 7 discuss the details of this approach in further details for flexibility strategies and learning strategies.

1.2 Situating this work

The new type of language tutoring system proposed in this dissertation can best be situated in the field of Intelligent Computer-Assisted Language Learning (ICALL), that brings Artificial Intelligence techniques into the field of Computer-Assisted Language Learning (CALL). As Melissa, Maisano, Alderks, and Martin (1993, p. 28) state: ICALL "might be more accurately described as parser-based CALL, because its 'intelligence' lies in the use of parsing - a technique that enables the computer to encode complex grammatical knowledge such as humans use to assemble sentences, recognize errors, and make corrections". This dissertation also has an error correction component embedded in the use of flexibility strategies as part of the meta-level architecture. Correcting errors is useful to provide immediate feedback to the learner but can also be used in the longer run to adapt future exercises to match the learner's level, estimated from the type of errors that he makes. This section positions the current work in ICALL by looking at other existing approaches to error correction and adaptivity.

Generally speaking, research into ICALL often emphasises one of three types of results: (1) the pedagogical effectiveness of the training, (2) the impact on the learner's motivation through individualised language instruction or (3) the performance of the technology that was used (Bodnar, 2012). This dissertation belongs to the third camp and focuses on the implementation of a new framework for future ICALL applications. It can already demonstrate a first proposal for error correction but the learner adaptivity of the framework currently remains theory.

1.2.1 Grammar formalisms

According to Matthews (1993, p. 5), when choosing a grammar formalism for a language tutoring application, one should consider a "formalism that potentially meshes with SLA". With the choice for Fluid Construction Grammar as a potential ICALL formalism, I follow Schulze and Penner (2008) who claim that Construction Grammar is a valid ICALL formalism that meets the three "criteria of adequacy" of Matthews (1993): linguistic perspicuity (everything is a construction), computational effectiveness (construction unification) and acquisitional perspicuity (applied in second language teaching, see for instance (Achard, 2008)). Moreover, they claim that the choice for a Construction Grammar formalism "will yield more adequate analyses of learner language and thus

not only improve, for instance, scaffolding and feedback for the learner but also provide new insight into characteristics of learner language and into language learning processes" (Schulze & Penner, 2008, p. 428).

Yet, I have no knowledge of any previous attempts of using a Construction Grammar-based computational formalism for ICALL. A range of other grammar formalisms have been used in ICALL from its start in the late 1980s, which have not only been employed and tested "when applying NLP to CALL, but also in many other projects which relied on computational linguistics in general" (Heift & Schulze, 2007, p. 61). Many of the early projects made use of Augmented Transition Networks, a procedural formalism that could be used to investigate transitions from one word to the next and used semantic and syntactic information on the arcs between nodes (Weischedel, Voge, & James, 1978). provides a list of grammar formalisms that have been used in ICALL since the early days until recently, of which I included the largest projects here below (cited by Heift and Schulze (2007)):

- Generalised Phrase Structure Grammar (Menzel, 1988, 1990, 1992; Heinecke, Kunze, Menzel, & Schröder, 1998; Menzel & Schröder, 1998, 1999).
- Head-Driven Phrase Structure Grammar (Brocklebank, 1998; Heift, 1998, 2001, 2002, 2003; Schulze, 1999, 2001; Heift & Nicholson, 2000; Heift & Schulze, 2003a)
- Lexical Functional Grammar (Feuerman, Marshall, Newman, & Rypa, 1987; Levin, Evans, & Gates, 1991; Levin & Evans, 1995; Rypa & Feuerman, 1995; Delmonte, 2002, 2003; Reuer, 2003)
- Tree Adjoining Grammar (Abeillé, 1992; Kakegawa, Kanda, Fujioka, Itami, & Itoh, 2000)
- Principles and Parameters approach (Weinberg, Garman, Martin, & Merlo, 1995; Hamel, 1996; Schulze & Hamel, 1998; Vandeventer, 2000, 2001; Vandeventer & Hamel, 2000; L'Haire & Vandeventer Faltin, 2003; Vandeventer Faltin, 2003)

1.2.2 Error correction

Once a full target grammar is implemented in Fluid Construction Grammar, it becomes relatively straightforward to carry out error correction thanks to the accuracy of the grammar's description. When a particular construction cannot apply that should have been applied, we can investigate exactly which feature in the construction caused the mismatch, which is probably the one that contains the error. Flexibility strategies are used

to interpret such errors and to suggest possible repairs so that the student utterance can still be parsed. Such flexibility strategies are typically general so that they can be used for different types of error (number, person, tense, verb class, etc.). There are typically two types of approaches to "analyzing a string with the goal of diagnosing learner errors", as has been nicely put by Meurers (2012): the mal-rule approach and constraint relaxation.

The mal-rule approach uses standard parsing algorithms that use mal-rules in addition to standard native language grammar rules to license learner errors (cf. Derek Sleeman (1982), Matthews (1992), as cited by Meurers (2012, p. 4)). Such mal-rules contain typical errors made by learners, and a new rule for every error needs to be instantiated. These rules are applied in parsing as soon as the standard rules fail to apply. Mal-rules have often been criticised because they do not allow for a scalable error analysis as every rule is manually defined. The alternative approach, known as constraint relaxation (Foth, Menzel, & Schröder, 2005), tries to "eliminate certain constraints from the grammar, e.g., specifications ensuring agreement, thereby allowing the grammar to license more strings than before" (Meurers, 2012, p. 4). A probabilistic use of constraint relaxation, with weights that control the likelihood of an analysis, can be useful to rank errors for a particular learner given a particular task (Foth et al., 2005). There are also some proposals of combining constraint relaxation and mal-rules. As Meurers (2012, p. 4) states: "Reuer (2003) combines a constraint relaxation technique with a standard parsing algorithm modified to license strings in which words have been inserted or omitted, an idea which essentially moves generalizations over rules in the spirit of meta-rules into the parsing algorithm".

1.2.3 Adaptivity

Although the advantages for adaptivity could not be demonstrated yet in this dissertation, I believe that the approach that I present here for an agent-based framework for ICALL applications certainly has future potential to create adaptive learning environments thanks to the combination of accurate language processing, agent-based modelling and meta-level strategies. A good overview of existing approaches to adaptive instruction is provided by Vandewaetere, Desmet, and Clarebout (2011), who distinguish between the source of adaptation (*to what will instruction be adapted*), the target of adaptive instruction (*what instruction will be adapted?*) and pathways of adaptive instruction.

1.3 Towards an artificial language tutor

The main foundations that this dissertation is built on all depart from the idea that successful and effective language learning is more likely to occur in natural environments

that lead to self-exploratory learning and with supportive help of an involved tutor who steers challenges and feedback in an optimal direction. Although the results that have been achieved in the work presented in this dissertation do not yet mirror this powerful idea, I hope to lay the basic building blocks of a new framework for designing ICALL systems. The three main elements of this new framework have been proposed above and will be developed further in the remainder of this section: a competent language agent, a language learning agent (student model) and tutoring strategies. This dissertation describes the design of these basic building blocks and demonstrates a first attempt at implementing a language agent and a student agent that interact in language games. Future development cycles will need to consider improvements of both models in terms of the representation of semantic information and the power of flexibility and learning strategies to take into account this information.

This dissertation describes the design of the basic building blocks of a new framework for ICALL and demonstrates a first attempt at implementing a language agent and a student agent that interact in language games.

Box 1.1 – Goal of the current work.

1.3.1 Modeling a competent speaker

Learning is generally more effective when a language teacher really speaks the language that he or she is teaching, especially when this is his or her first language. When your French teacher really lived in France and persists in using French as the only language that is used in class, you are much more likely to pick up the language. By contrast, most of those that ever studied Latin will probably not remember much more of the language than some rote-learned declension paradigms such as *rosa, rosam, rosas, rosae, etc.*. Therefore, when designing an effective language tutoring system it is necessary that the artificial tutor itself is a competent speaker of the language that is being taught by the system. As a proficient language user, the tutor must thus be capable of conceptualizing and produce utterances in context or to parse and interpret them. This dissertation introduces the notion of a *language agent* to fulfill this need. A language agent represents an ideal speaker of a language whose linguistic skills also allow him to correct erroneous utterances of others.

The language agent that is presented here consists of three main components: a construction inventory, a grammar engine and a set of learning strategies (see Figure 1.1 for a visualization). The first component, the construction inventory, is a catalogue of all the grammatical constructions that language user typically uses. It can contain lexical constructions, phrasal constructions, morphological constructions, etc., that are

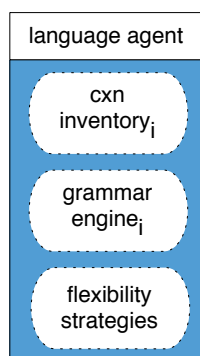


Figure 1.1 – A language agent functions as a model of a language speaker through actively producing and parsing utterances. It has a construction inventory (grammar), a grammar engine that can process constructions and a set of flexibility strategies.

each responsible for a small part in the processing of an utterance. The construction inventory can be organized according to different principles that are either driven by the implementation and processing perspective or by the psycholinguistic relevance of grammar organization.

The second main component is the grammar engine. This is the component that is responsible for the actual linguistic processing of the constructions that are collected in the construction inventory. Such processing involves a search through the inventory to retrieve the constructions that are required to build or interpret a particular utterance. Moreover, the grammar engine should allow for bidirectional processing so that the same constructions can be used in production and parsing. This bidirectionality is a crucial feature if we want to enable flexible processing, which implies that the tutor can try to reproduce the student's utterance to reconstruct the constructions that he accessed and the possible search path that was taken.

Finally, a language agent also has a set of flexibility strategies that are in responsible for the open-endedness of the linguistic processing of learner utterances. Ungrammatical forms need to be captured and treated accordingly without any visible disruption of the interaction between tutor and student. This specification requires that the language processor can on the one hand detect ungrammatical input and on the other hand once a potential error is retrieved, suggest a correction that is most plausible in the current situation and continue regular processing. This notion of repairing a problem "behind the scenes" echoes the ideas of computational reflection in the 80ies, which are still being used to today's operating systems and programming languages (Maes, 1988; Maes & Nardi, 1988; B. C. Smith, 1982). I employ this notion here through the use of *diagnostics* and *repairs* that operate during default constructional processing. Diagnostics merely signal a (potential) problem, whereas repairs can modify data in the temporary processing pipeline so that a solution can be found. They do not change anything internal in the agent.

1.3.2 A predictive student model

A good teacher naturally constructs a model of his student that represents the student's skills and knowledge as a function over time. It is a kind of model that could mimic typical student utterances that are illustrative of the student's proficiency level. In order to operationalize such a predictive model it is convenient to reuse the three-component language agent architecture. This student model is thus implemented as a fully fledged agent, who can actively participate in the linguistic community that he finds himself in. This agent is further also referred to as *a student agent*.

Because the language agent's and the student agent's architectures are identical (see Figure 1.2), it becomes very cost-efficient to construct a student model from scratch. The most important difference is, of course, the difference in competence level between the tutor and the student. The student does not yet master all the constructions that are needed to be fully expressive in the language that he is learning. Gradually, his construction inventory will expand and mold itself towards the target language. It might take different paths to construct an L2 language, so that different learning strategies are required. As a result, the realization of the language agent and the student agent architecture's components is not identical, only the components themselves are homologous.

Learning strategies encode personal tactics on how to solve a particular problem and they can thus differ greatly between students. For instance, one learning strategy for learning Catalan would be to first conjugate all the verbs in their first person singular form. Another strategy could imply that you construct your sentences in Spanish (in case you master this language) and replace some of the words by their Catalan counterparts. However, the current implementation of learning strategies does not yet go this far in mimicking strategies that learners use but is rather a proof of concept that learning strategies are a valid method to learn a target grammar from input sentences that achieve

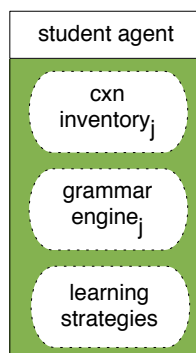


Figure 1.2 – A student agent functions as a student model and can be used by the tutor to actively predict the real student's skills. It shares an identical structure with the language agent, which is convenient to model changes in linguistic knowledge and learning strategies.

100% accuracy.

1.3.3 Tutoring strategies

Apart from making a dynamic model of his students, a human teacher typically also applies a range of tutoring strategies to assist students in their problem-solving tasks. A tutoring strategy is a dynamic plan of action that stipulates future interactions with the student. To create or adapt a tutoring strategy, a teacher does not only depend on the information that is kept in the student model but he also makes use of a more general record of the student's strengths and challenges in learning. Although this dissertation makes first suggestions for using tutoring strategies in the agent-based tutoring architecture that is proposed, it is not supported by a working implementation. A first prototype implementation is however within reach thanks to the reuse of the meta-level strategies for tutoring and its heavy reliance on the language and the student agents.

The language tutoring system that is proposed here hosts an artificial tutor that simulates these typical teacher tactics. As a result, the original language agent architecture needs to be extended so that this agent can also function as a tutor (see Figure 1.3a). Such a revision implies two new components to that are part of a tutor agent, apart from having direct access to the student agent: a tutoring strategies component and a student profile component. These components are vital elements of a personalized tutoring approach because they provide meta-information about the tutoring process, for instance to decide which type of exercise to repeat or where to challenge the student further.

The tutoring strategies component is comparable to the set of learning strategies that is present in the language agent. However, they now comprise strategies that can be used in the process of tutoring a language to a less proficient language speaker. Strategies can differ widely across tutors and they depend on the goals of the student. For instance, a German-speaking colleague could decide to teach you only German words and sayings that are really used by German language users, rather than more normative ways of speaking. Also the constructions that you want to teach to a student might vary to a great extent depending on his learning motivation. A student that needs to learn English in order to be able to read scientific articles on genomics has to learn a different vocabulary than someone who wants to go salmon fishing in Scotland.

The second component that a tutor agent needs is a student profile. A student profile is a record where a tutor keeps a full log of the student's activities and summaries on his progress and recent performance. It can be used by the tutor's tutoring strategies to set the right challenge level for the student. This record is a personal account that belongs to a particular student and does not allow for any direct comparisons between students that are being tutored by the same or - certainly not - by different tutors. The

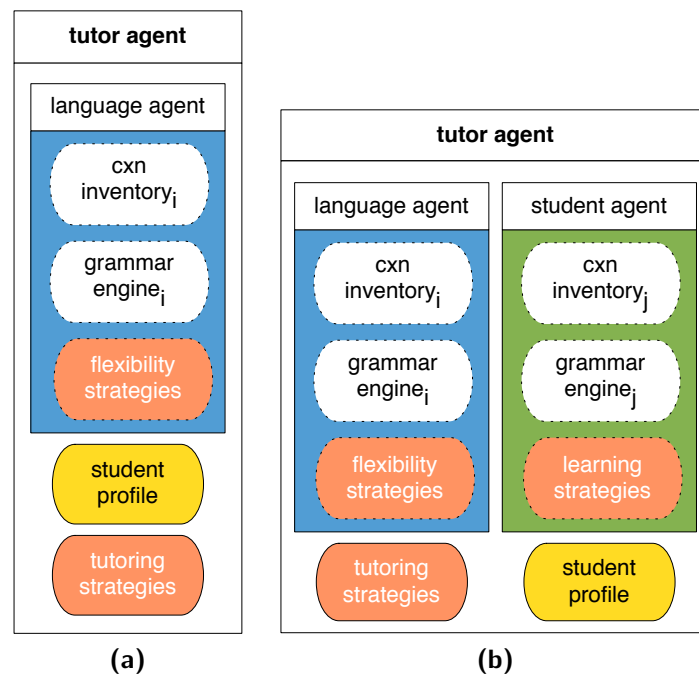


Figure 1.3 – (a) A language agent can be extended with a tutoring strategies component and a student profile component to become a tutor. These components personalize the tutoring process by keeping essential information about the student that is constantly being updated so that tutoring can be personalized to better fit the motivations of the individual student. (b) The tutor agent should have direct access to the student agents to adapt its tutoring strategies to the level of the student.

main function of the student profile is to inform the tutoring strategies component on whether to modify a strategy or add a new one.

Finally, the tutor agent has immediate access to the student model so that he can scrutinize the actual state of the agent's construction inventory and learning strategies (Figure 1.3b). It is only then that he can properly align the student model to the real student that is being coached. The student model is aligned after every interaction that the student has with the tutor agent. When the student was successful in the current task, the tutoring strategies will indicate how to update the student agent's constructions that correspond to the task. And also when there was a failure or mismatch between the goals of the tutor agent and the real student, the student agent proves useful to verify whether the mismatch could have been predicted based on the student model or not. Because the construction inventories have the same architecture in the tutor and the student agent, this symmetry can be used to learn about possible gaps or inconsistencies in the student's grammar.

1.4 Structure of the dissertation

This dissertation is structured as follows: Part 1 provides more background information to situate the current work in the field of Intelligent Computer-Assisted Language Learning (ICALL), and more generally Artificial Intelligence in Education. Readers that have a background in Intelligent Tutoring Systems for language can skip this chapter. Part 2 describes the operationalization of the language agent and the student agent, two basic building blocks of the language tutoring framework that this dissertation presents. Part 3 includes suggestions for future development of the student agent to become an effective student model and proposes a first design of a tutoring game with a human student in the language-game loop. The final part concludes this dissertation by recapturing its main contributions and pointing to questions for future research. Part 2 consists of the following six chapters:

Fluid Construction Grammar How is grammatical knowledge represented in the language agent? This chapter provides answers to this question by showing different types of constructions in language, how they are created and how they are organised within a grammar. Also the workings of the grammar engine that processes constructions is described and the concept of constructional search is introduced. Advanced users of Fluid Construction Grammar (FCG) can skip this entire chapter.

Spanish verbs in FCG The complexity of Spanish verb conjugation can be fully captured by different types of constructions in FCG: lexical, phrasal, morphological and phonological. This chapter provides illustrative examples of each of these constructions

and demonstrates how individual verb forms can be processed (parsing and production) with the right grammar engine settings. The differences in processing between regular, semi-regular (part of the paradigm is irregular) and irregular verbs are highlighted.

Flexibility strategies This chapter investigates how a language agent can reach a higher linguistic competence level once his grammar is in shape. Flexibility strategies are an additional tool a language agent might use to diagnose and analyse the slightest deviations that occur during processing. Using the powerful ideas of meta-level architectures and reflection, flexibility strategies put the routine processing pipeline on a hold and try out solutions for the diagnosed problem in a meta-layer. Once a valuable solution is found, processing can be restarted at the same point where we left off or at an earlier point in the pipeline. It is this type of strategies that will be used for error analysis and correction in the artificial language tutor.

A language agent for Spanish verbs This final chapter brings all the processing elements together into a fully operational language agent that can conjugate any Spanish verb form and detect and repair verb form errors made by learners. Once the specific flexibility strategies for Spanish verbs are introduced, the verb form error correction is evaluated by a corpus-based error analysis on the Spanish Learner Language Oral Corpora (SPLLLOC). The language agent has to parse erroneous verb forms from the corpus and restore them to an acceptable correction. When the language agent's corrections are compared with those made by a human teacher, 77% of all corrections overlap.

Learning strategies Inspired by the same ideas as the flexibility strategies of a language agent, a student agent is endowed with learning strategies to solve problems that occur during processing. Yet, because his grammar is not yet mature enough, learning strategies target learning problems and repair them by adding a new construction to the grammar or adapting an existing one. Also grammar engine settings can be modified during learning. Learning strategies work on different levels of the construction inventory and can be scored using a weighting system.

A student agent for Spanish verbs This chapter describes the learning strategies that are needed to acquire Spanish verbs through situated interactions with a competent language agent. The effects of these strategies on the learning of the full conjugational paradigms of 25 irregular, semi-regular or regular verbs is demonstrated. As an alternative - and much faster - way to bootstrap the Spanish construction inventory needed to conjugate any verb in Spanish, I introduce a decision tree classification algorithm that automates the verb paradigm construction creation for any Spanish infinitive. This approach was inspired by the online conjugation tool for Spanish *Onoma* (www.onoma.es).

Part I

Background

Chapter 2

Artificial Intelligence in Education

As soon as primary and secondary school class sizes began to increase and machines started to take up an important role in people's daily lives, the idea of a tutoring machine that could provide individualized instruction became more popular. The behaviorist B. F. Skinner presented the first "teaching machine" in the 1950ies, in the form of an incremental mechanical system that would reward students for correct responses to questions (Skinner, 1961). The idea was later reinforced by the famous two-sigma problem that could show that student achievement in classroom interaction differs greatly from results obtained from individual tutoring (by about two standard deviations) (Bloom, 1984) (Figure 2.1). Because most schools cannot provide this individual attention, information technology is often a valid alternative for human teachers.

Although the supply of commercial intelligent tutoring systems (ITSs) is very large today, especially when you take into account the recent rise of web-based tutors, most of them do not offer the individualised tutoring that they promise but instead stick to the old idea of programmed instruction, where every student runs through the same curriculum in the same order. Contrasting to these static learning environments where the same information, the same structure and the same interface is presented to every individual student, adaptive learning environments are currently on the rise and pose new possibilities and challenges to the field. Wauters, Desmet, and Van den Noortgate (2010) mention three dimensions of adaptivity: the form (adaptation techniques for form or content representation), the source (features involved in the adaptivity process) and the medium (adaptive hypermedia or intelligent tutoring systems).

Even if the final goal of the framework presented in this dissertation is to build an adaptive tutoring system in the long run, the current results illustrate how the agent-based architecture can accomodate for both a natural language processing component that allows to analyse learner language accurately (language agent) and a learning component

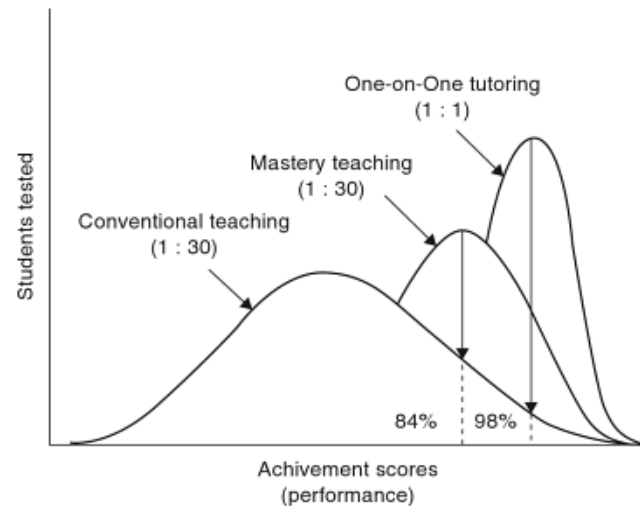


Figure 2.1 – The advantages of one-to-one tutoring become clear when one compares the average achievement scores of conventional teaching and one-on-one tutoring. The average achievement scores in a tutorial setting are centered around the 98% percentile of the conventional teaching scores. The difference between mastery learning and the conventional setting lies in the kind of testing: periodical tests for determining students' mark vs. formative tests for the purposes of feedback followed by corrective procedures (Figure reproduced from (Park Woolf, 2008)).

that demonstrates how the target system can be learned by an agent that starts with no previous linguistic knowledge. These two components are invaluable building blocks for a future adaptive tutoring system. The current chapter provides further background on the history of Artificial Intelligence in Education.

2.1 A brief history of Intelligent Tutoring Systems

Although today a well-established concept, Intelligent Tutoring Systems (ITSs) have gone a long way since the first breakthroughs in the early seventies that incorporated AI techniques into programmed instructions. These early advances allowed for (i) alternative representations of content, (ii) alternative paths through material and (iii) alternative means of interaction. Much of the research into expert systems turned out to be useful for representing expert (tutor) knowledge and building student models. In the eighties, and still very much so today, the main research questions of the field of ITS could be formulated as follows (Self, 1988):

- What is the nature of knowledge, and how is it represented?
- How can an individual student be helped to learn?

2.1. A BRIEF HISTORY OF INTELLIGENT TUTORING SYSTEMS

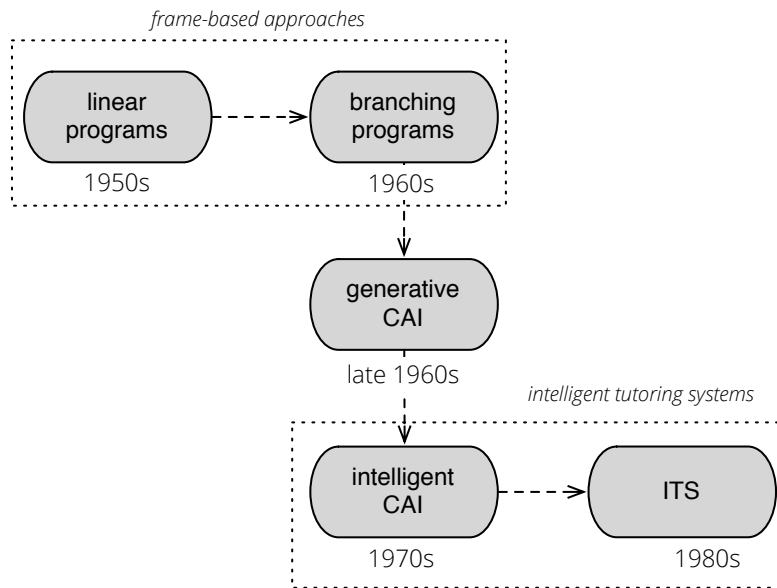


Figure 2.2 – The history of Intelligent Tutoring Systems

- Which styles of teaching interaction are effective, and when should they be used?
- What misconceptions do learners have?

The current section describes the history of ITSs and shows how they evolved out of programmed instruction applications (see Figure 2.2 for an illustrative diagram of this evolution) in Section 2.1.1. Also the rise of ITSs for language learning, or *Computer Assisted Language Learning* (CALL) is discussed further in Section 2.1.2. Finally, Section 2.1.3 describes current trends in the field, with a special focus on CALL and the recent rise of massive open online courses (MOOCs).

2.1.1 From frame-based approaches to ITSs

The first attempts to build tutoring systems were all **frame-based**, where most frames contained simple questions (fill the gap exercises, selecting the correct answer, etc.). Such tutoring systems proceeded to present the next frame regardless of the accuracy of the student's response. It was therefore nothing more than a programmed text book, completely lacking any individualization. In the 1960ies, Crowder tried to overcome this major shortcoming as he introduced the notion of **branching programs**. Although still having only a number of fixed frames, these programs no longer ignored student's responses but the system could comment on a student's response and use it to choose the next frame (Crowder, 1964) (Figure 2.3).

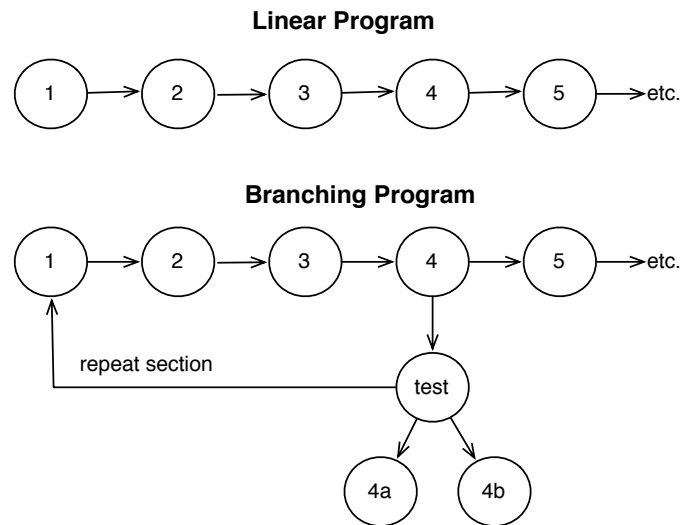


Figure 2.3 – Whereas linear programs provide only a single trajectory through a tutoring session, branching programs provide subtests on particular tutoring steps and also allow for repetition.

Generative Computer-Assisted Instruction (CAI) was launched in the late 1960ies. The idea of **generative CAI** was that teaching material could be *generated* automatically by a computer. One of the main advantages was that memory usage could be considerably reduced by this technique, since the frames did not have to be saved as such. However, this approach remained restricted to drill-type exercises, in which the learner model consisted of nothing more but an integer. Uhr and his collaborators (Uhr, 1969) implemented a series of systems which auto-generated problems in vocabulary recall and arithmetic, two domains which presumably require drill and practice types of exercises. The sophistication in their systems was situated in the task-selection mechanism, which ensured the exercise level to be adapted to the student's overall performance.

It was Jaime Carbonell's mission to **put AI into CAI**, meaning that the computer should have a representation of *what* is being taught, *who* and *how* (Carbonell, 1970b, 1970a). He developed SCHOLAR, a tutoring system for teaching Latin-American geography. SCHOLAR helped students enhance their knowledge by (i) solving problems at a certain level or by (ii) involving them in discussions with the computer in a more interactive way. The following description captures the essence of the Scholar tutor:

In the transition of CAI to ITS, the basic methodology of Scholar was a tutorial dialogue using templates and keyword recognition. It was used for teaching Latin American geography through inquiries and answers on a topic randomly chosen by Scholar. This man-to-machine tutorial system enables to individually help students enhance their knowledge by solving problems at a certain level or by involving in discussions in a more interactive way with the computer (Bruce, n.d.).

2.1. A BRIEF HISTORY OF INTELLIGENT TUTORING SYSTEMS

Although there is no sharp boundary, in the 1980ies intelligent CAI was replaced by **Intelligent Tutoring Systems** (ITS), that try to extend the domain of applicability, power and accuracy of CAI systems (Clancey, 1979; Self, 1974, 1977, 1985). Examples of early ITS include the Pittsburgh Urban Math Project (PUMP) algebra tutor (Anderson, Corbett, Koedinger, & Pelletier, 1995) and the SHERLOCK control panel (Lesgold, Lajoie, & Bunzo, n.d.), used to train Air Force technicians to diagnose problems that might occur. Another early system is GUIDON (Clancey, 1979, 1987), which was the first intelligent tutor that is based on an expert system. GUIDON was also the first program to teach medical knowledge. The project became relevant in developing future medical tutoring systems because of some key insights: "the need to represent implicit knowledge and the challenges of creating a knowledge representation sufficiently large, complex and valid to help students learn real medical tasks" (Park Woolf, 2008, p. 18).

2.1.2 Language learning tutors

Although some of the first tutoring systems (PLATO; (Hart, 1981)) already contained some form of natural language dialogue with their users, real language tutoring systems (i.e. more than just vocabulary drill) only came in vogue in the early nineties, when some AI technologies "were mature enough to be included in language learning systems, at least in experimental settings" (Gamper & Knapp, 2002). Building an artificial tutor that can teach language is very different from guided exercises in mathematics or physics. Language education brings along a level of complexity that is different from problem solving domains. A language tutor's domain knowledge is a qualitative representation of grammatical knowledge, which is an analytic and unverifiable domain.

According to Levy (1997, p. 1), Computer-Assisted Language Learning (CALL) can be defined as "the search for and study of applications of the computer in language teaching and learning". The use of AI in CALL is also referred to as intelligent CALL or 'ICALL'. For others, ICALL is even more specific because "it might be more accurately described as parser-based CALL, because its 'intelligence' lies in the use of parsing - a technique that enables the computer to encode complex grammatical knowledge such as humans use to assemble sentences, recognize errors, and make corrections" (Melissa et al., 1993, p. 28). However, to avoid all confusion, I prefer to use only a single name to refer to language learning that is assisted by an artificial tutor and I will therefore consistently use the term CALL as the more general term that encompasses all of its "intelligent" variants.

To understand the history of the field, we have to go back to the field's first seminal publication the book *Intelligent Tutoring Systems for Foreign Language Learning*, edited by M. L. Swartz and M. Yazdani in 1992. In this book, "several intelligent methods such as grammar checking, error analysis, user modeling, and tutoring are discussed and how

they can be adapted and combined to be useful for language learning systems" (Gamper & Knapp, 2002, p. 330). The first years of CALL research had been very much focused towards the development of technologies and less on the pedagogical side of it. More concretely, this meant an article that was entitled "Computer-Assisted ESL Research" would actually investigate "how simple for-next loops can be creatively integrated into different programs to act as timers" (Brownfield, 1984, p. 20). Yet, in 1994 a special issue of the renowned *International Journal of Artificial Intelligence in Education* was completely dedicated to the topic of language learning. "Besides the technological aspects, researchers begun to include results from pedagogy and cognitive science as well as to consider teaching pragmatics and socio-linguistic competence" (Gamper & Knapp, 2002, p. 330). Yet, at that time, none of the systems presented in the issue was used in real learning situations.

Another important publication to understand the evolution of the field is the volume *Intelligent Language Tutors – Theory Shaping Technology* that was issued in 1995. The systems that were presented in this volume stressed the usefulness of NLP techniques that can provide "a more comprehensive language teaching environment including negotiations and discourse between the learner and the system" (Gamper & Knapp, 2002, p. 330). In the following years, as research in Automatic Speech Recognition (ASR) and NLP matured so that these techniques can be fully integrated into a CALL system architecture. Matthews (Matthews, 1993) argued that it is possible to "conceive of an ICALL system in terms of the classical ITS architecture" (see Section 2.2), consisting of three parts: an expert, student and a teacher module. The expert module is then seen as the module that "houses" the language knowledge and, ideally, it is this part that can process any piece of text produced by a language learner (Heift & Schulze, 2007, p. 2).

The research field of CALL is still very active today, with three main journals that are dedicated to its objectives (*Computer Assisted Language Learning*, *CALICO* and *ReCALL*) and some major conferences that brings together researchers in the field (WorldCALL, EUROCALL, CALICO).

2.1.3 A third generation of ITS research

We can now speak of at least two generations of ITS research, and we are currently at the dawn of a third one, which will probably be much more revolutionary. The first generation spans roughly from 1970 until 1990, a period of thirty years in which more than 40 systems were released. This early generation was powered by the booming of Artificial Intelligence, a field that was seeking applications for its technologies. Moreover, CAI seemed a mature and promising technology and solutions had to be found for the increasing group sizes in schools. The second generation, ranging from roughly 1990 until today has formulated the scientific foundations of the field (Self, 1990) and witnessed

2.1. A BRIEF HISTORY OF INTELLIGENT TUTORING SYSTEMS

the launch of the *International Artificial Intelligence in Education* (IAIED) journal, the biannual AIED conference and the biannual ITS conference. Also implementations of real systems in schools realized, thanks to new spread of digital technologies in traditional education.

The remainder of this section outlines some of the most current trends in language learning tutoring and describes the recent phenomenon of the MOOCs, allow for large-scale online tutoring.

Current trends in CALL

Some of the current trends in the field of CALL include, among others: the use of social media and networking, multimodality, mobile learning, virtual learning environments, distance and collaborative learning and the question of assessment and feedback. All of these topics (and many more) were addressed at the 2012 Eurocall conference in Gothenburg. A recent special issue in the ReCALL journal explored the challenges and opportunities of digital games for language learning (Cornillie, Thorne, & Desmet, 2012), dealing with issues such as the perception of corrective feedback, the linguistic complexity used in games, etc. Also Mobile-Assisted Language Learning (MALL) is gaining ground as a platform for self-regulated learning (Sandberg, Maris, & de Geus, 2011). The simple practice of MALL would already foster an advanced form of self-study and stimulate self-regulated learning in which students take responsibility for triggering and sustaining their own motivation.

Affective computing is another important trend that is influencing the development of new intelligent tutoring systems. D'Mello and colleagues detected three main emotions that play a role in tutoring: boredom, frustration and confusion (Baker, D'Mello, Rodrigo, & Graesser, 2010). He measured those feelings (without interrupting the tutoring process) based on two methods: (i) facial-expression recognition software and (ii) the use of a special chair with posture sensors that could "tell whether students are leaning forward with interest or lolling back in boredom" (Murphy Paul, 2012). As a less expensive alternative, the emotions of a student can also be judged based on the pattern that manifests itself in his answers to the tutor's questions. This research topic is investigated by "educational data mining" techniques that help to determine characteristic student answers (Pardos, Gowda, Baker, & Heffernan, 2012; Walonoski & Heffernan, 2006).

The year of the MOOC

Tutoring systems have recently scaled up dramatically in size, with the year 2012 announced as the year of the MOOC (Pappano, 2012), short for Massive Open Online

CHAPTER 2. ARTIFICIAL INTELLIGENCE IN EDUCATION

Course. The first MOOC that really approached "massive" student numbers was a free course on Artificial Intelligence, CS211, offered by the University of Stanford in the Autumn of 2011. The course tutors, Sebastian Thrun and Peter Norvig, attracted 160,000 registered students from 190 different countries. Although students would not receive any official credits for completing this course, 20,000 students (that is, 1/8 of all students enrolled) finished the course and were sent a PDF of their "statement of accomplishment". Thrun and Norvig recently launched a start-up company called Udacity that delivers similar free online courses. Other initiatives that have appeared since the success of the CS221 MOOC include Coursera (an educational company founded by Stanford professors Andrew Ng and Daphne Koller) and EdX (a joint partnership between MIT and Harvard).

The ultimate dream of designing a MOOC include ideas such as "loosening institutional control of learning outcomes and assessment criteria, shifting from a focus on content delivery to a foregrounding of process, community and learning networks, and working with more exploratory assessment methods digital and multimodal assignments, peer assessment and group assignments" (Knox, Bayne, Macleod, Ross, & Sinclair, 2012, p. 3). However, the reality often looks very different: multiple choice questions, numerical answers, short text answers, structured outputs and peer assessment (Ng & Koller, 2012, p. 4).

The potential of MOOCs is enormous but there are still some issues in designing and running MOOCs. The most unsettling issue for many educators running an open course is the high dropout rate (Cormier & Siemens, 2010). As the following account by Siemens (n.d.) testifies: "While active participation in our courses declines as the course progresses, subscribers to the Daily increase. I'm not sure what to make of that. If I was getting five emails a week on something I wasn't interested in, I would unsubscribe. Does that mean we can view Daily subscribers as a) people are still engaged, b) people can't find the unsubscribe link, or c) that we've subjected over 15,000 people to guilt about not being active in c-MOOCs?". Some critics have questioned whether MOOCs are at the forefront of the "McDonaldization" of higher education (Lane & Kinser, n.d.). By enabling a few elite institutions to broadcast "their star courses to the masses", they foster little engagement or cross-cultural understanding. The "massive" element of MOOCs might have a homogenizing effect to education across the globe.

2.2 The general ITS architecture

As a result of the developments described in Section 2.1, an ITS has a standard architecture today with a number of components that are each responsible for a specific function. The components can best be explained according to the knowledge type they

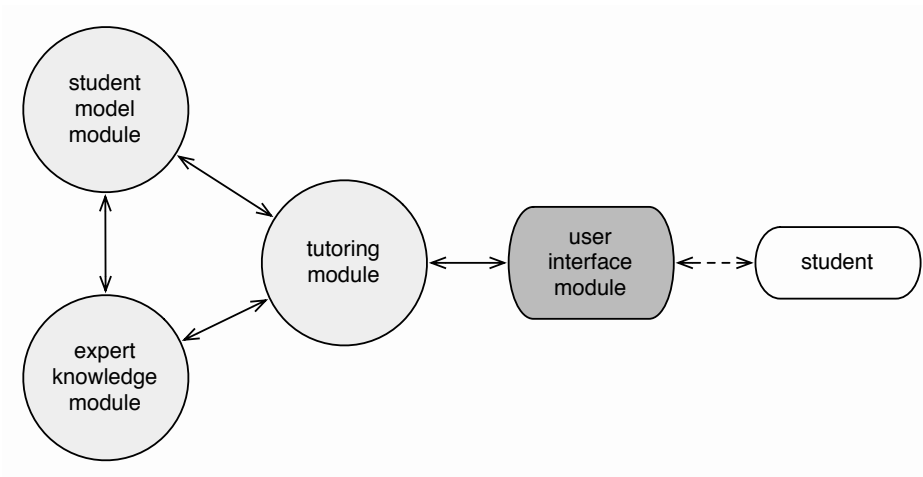


Figure 2.4 – The basic architecture of an Intelligent Tutoring System consists of three main modules:

encode, which results in the following four types:

1. **Domain knowledge** (how experts perform in the domain):
definitions, processes or skills needed to multiply numbers (e.g. the AnimalWatch tutor), generate algebra equations (e.g. PAT tutor), etc.;
2. **Student knowledge** (how to reason about student knowledge):
stereotypic student knowledge of the domain and information about current student (time spent on problems, hints requested, correct answers, preferred learning style, etc.);
3. **Tutoring knowledge** (encoding reasoning about the feedback):
either derived from empirical observations of teachers or enabled by technology (simulations, animated characters);
4. **Communication knowledge**:
includes graphical user interfaces, animated agents, dialogue mechanisms.

The domain knowledge module (expert knowledge), the student model module (student knowledge) and the tutoring module (tutoring knowledge) are interconnected in the main architecture of an ITS. Communication knowledge is incorporated by a user interface module, that mediates between the student input and the tutoring module (Figure 2.4). Because the communication knowledge is often included in the tutoring module, the remainder of this section discusses the three main interconnected modules in the ITS architecture: expert knowledge (Section 2.2.1), student knowledge (Section 2.2.2) and tutoring knowledge (Section 2.2.3).

2.2.1 Expert knowledge

Domain models interact very closely with the student model: they are the first step in representing the expert knowledge. They can generally be divided into three categories of complexity: (i) problem solving (mathematics problems, Newtonian mechanics), (ii) analytic and unverifiable domains (ethics, law) and (iii) design domains (architecture, music composition). There are two main axes in the classification of domain models: a first one ranging from simple to complex and a second one ranging from well-structured to ill-structured. Category 1 models represent expert knowledge in the field of arithmetics and other well-defined domains (well-structured, simple). Category 3 represents the other side of the axes: complex and ill-structured domains such as the knowledge needed to build an ITS for architecture tutoring. Finally, category 2 contains qualitative representations of expert knowledge for fields such as language, which are halfway on both axes (Park Woolf, 2008).

Linguistic knowledge is often classified as a category 1 domain model, for which the typical teaching strategy is to present a battery of training problems or tests (Lynch, Ashley, Aleven, & Pinkwart, 2006). Student's work is typically checked for correctness. Yet, defining all rules of a particular linguistic system such as verb conjugation, including all student misconceptions, is a difficult and time-consuming task that often needs to be carried out by hand.

Although linguistic knowledge is classified as a well-structured domain, it in practice requires big efforts to formulate all rules of a linguistic system and incorporate all student misconceptions.

Box 2.1 – Linguistic expert knowledge

2.2.2 Student knowledge

A student model can be defined as the set of beliefs that a tutor has about a student. These beliefs include the knowledge and skills of the student in the target domain, his learning preferences and other attributes. They can be inferred based on a student's observable behavior: through his answers, actions or the results that he obtains. Traditional ITSs keep track of a student's performance based on a series of preset learning objectives, such as a range of grammatical phenomena in the target language or vocabulary items covering the learning situations that the student has selected.

To improve the student modeling enterprise, some tutoring systems allow their students to inspect and control the student model (Cook & J. Kay, 1994). Student models with

this property are called *open learner models* (OLMs). They can contain simple overviews of knowledge (such as a skill meter) or a more detailed representation of knowledge, concepts etc. Park Woolf (2008) lists several motivations for the use of open learner models, such as (i) the student has the right of access to and control over his personal information; (ii) the student can potentially correct the learner model; (iii) the frequent asymmetric relationship between the student and the tutor can be resolved; and (iv) OLMs stimulate reflective learning in the student.

Researchers in ITS tend to classify their student models according to three main dimensions. The first one covers the input that the system receives, while the remaining two are structural properties of the student profile. VanLehn (1988) refers to them as *bandwidth*, *target knowledge type* and the *differences between student and expert*. The remainder of this section discusses all three with respect to their use in existing ITS applications and relates their use to the domain of language tutoring.

Bandwidth

Bandwidth refers to the amount and quality of the input that the diagnosis component receives about what the student is doing or saying. From this input, the tutor must infer what the student is thinking and believing (VanLehn, 1988). Most variation between tutoring systems can be captured in terms of three levels of bandwidth information. The highest amount of information that the tutor can obtain is indirect information "that approximates the students' mental states". Of course, our mental states are not (yet) directly accessible by machines (or humans), so the full "mental states" bandwidth is not realistic to achieve. Yet, by "asking enough questions or by eliciting verbal protocols", the tutor can obtain enough indirect information to approximate the student's current *mental state*.

The remaining two bandwidth categories are *final states* and *intermediate states*. Sometimes the tutor can only see a student's answer (or final state) to an exercise, but in other situations such as when solving algebraic equations or when playing chess, also the intermediate steps that the student went through are observable by the tutor. According to VanLehn (1988), the bandwidth dimension is perhaps the most important dimension of a tutoring system since it "determines the algorithm used for diagnosis".

Bandwidth in language tutoring is usually restricted to final states of the student's answer to an exercise (production/comprehension).

Box 2.2 – Language tutoring bandwidth

In language tutoring we usually only see the student's final answer in the form of an

utterance without having access to the intermediate processing steps or the conceptualization that the student has made to reach this final step. If the tutor needs access to intermediate steps, they will have to be constructed through a simulation of the student's answer. Using an active student agent is the solution this dissertation proposes (see Chapter 9).

Knowledge type

Because a good student model can in practice solve the same problems as a real student would be able to solve, it can be used to actively *predict the real student's answer*. To solve these problems the model needs "some kind of interpretation process that applies knowledge in the student model to the problem" (Polson & Richardson, 2013, p. 60). Depending on whether we are dealing with *procedural* or *declarative* knowledge, a different interpretation process is required. The interpretation of procedural knowledge is less costly because it can rely on local search given the current state of the problem. For declarative knowledge, the whole knowledge base (or construction inventory) has to be searched to find a solution. However, the distinction between procedural and declarative knowledge is known as a rather fuzzy differentiation in AI. Many instances of problems are in fact a mix of procedural and declarative knowledge. For instance, the medical tutor GUIDON's knowledge of medicine (Clancey, 1987) is partly declarative (coupling symptoms to diseases) and partly procedural (determining which questions to ask the patient according to which circumstances).

Also in language processing language users rely on a mix of declarative and procedural knowledge. Irregular verb forms have to be accessed directly without any intermediate processing steps, while other verb forms are constructed through a range of local steps. However, the distinction between procedural and declarative, or the complexity of interpretation, defines the complexity of diagnosis. According to VanLehn (1988), declarative knowledge is easier to diagnose since there is only one step where things might have gone wrong. In procedural knowledge, many items could have been accessed, which complicates the diagnosis. Yet, although this distinction might be valid for programming tutors or medical tutors, I would disagree with his viewpoint for language tutoring. Having access to many subprocesses and perhaps failed branches in the local search of procedural problems might actually be beneficial for the diagnosis of a problem as a much finer-grained diagnosis becomes possible. By contrast, in the diagnosis of declarative problems diagnosis is often rather ad hoc and uncertainty values are higher as to whether the right diagnosis was made.

Having access to many subprocesses and failed branches in the local search of procedural problems might actually be beneficial for the diagnosis of a problem as a much finer-grained diagnosis becomes possible.

Box 2.3 – Procedural knowledge is more informative for diagnosis

Student-expert difference

The knowledge of a student is usually regarded as the background knowledge of a student modeling system. In traditional intelligent tutoring systems this knowledge is defined as:

- the correct facts, procedures, concepts, principles, schemata and/or strategies of a domain; and
- the misconceptions held and other errors made by a population of students in the same domain (sometimes referred to as the bug library).

The background knowledge may also contain historical knowledge about a particular student (e.g., past qualitative models and quantitative measures of performance, student preferences and idiosyncracies, etc.), and stereotypical knowledge about student populations in the domain (Sison & Shimura, 1998, p. 131).

Student knowledge always needs to be understood in relation to an expert model that can provide explanations on the correct way(s) to solve a problem. To compare student and expert or tutor knowledge, most ITSs claim to use the same knowledge representation language for both (VanLehn, 1988). However, reality is often different. Due to economy and other implementation issues, the student model is often a copy of the expert model plus a collection of differences: missing concepts (knowledge that the student does not yet have) and misconceptions (knowledge that the student has that the tutor does not). There are three major ways to represent differences between the student and the tutor model: overlay models, bug libraries or bug part libraries and constraint-based models. Recently, so-called model-tracing tutors have appeared, which try to "interpret and assess student behavior with reference to a cognitive model that can solve problems in the way that competent students can" (Aleven, McLaren, Sewall, & Koedinger, 2009, p. 107).

- **Overlay models** assume that student knowledge is always a subset of the tutor's knowledge (Carr & Goldstein, 1977). The student model consists thus of the tutor model plus a list of items that are (still) missing. Once the expert's knowledge has been enumerated in a number of rules or plans, overlay models are fairly easy to construct. Often, domain knowledge is annotated and each expert step is assigned

a particular weight. Missing steps in the student's knowledge can thus be scored accordingly.

- **Bug (part) libraries** are "a mechanism that adds misconceptions from a predefined library to a student model" (Park Woolf, 2008, p. 52). Misconceptions are thereby manually coded in so-called *mal-rules*. A problem is diagnosed by findings bugs from the library so that the student model (current student's knowledge state) can be constructed when these bugs are added to the tutor model. Yet, it is generally impossible to ever have a complete bug library. And even if we could construct such a library, "at the start, a bug library that contained at least the most common errors of a group of students, the results of Payne and Squibb (1990) suggest that different groups or populations of students (e.g., students from different schools) may need different bug libraries" (Sison & Shimura, 1998, p. 131). To make bug libraries less rigid and allow for more automatic bug extensions or creations, bug parts libraries were invented. These libraries only contain subparts of bugs and the real bugs are assembled dynamically during diagnostics. Although the problem of creating bugs and partial bugs are very similar, because libraries of bug parts are much smaller, problems become often easier to solve.
- **Constraint-based models** try to interpret and assess student knowledge with respect to a set of constraints that all student answers should satisfy (Mitrovic & Ohlsson, 1999). Formally, constraints are pairs consisting of a relevance part and a satisfaction part (Ohlsson, 1994). "Using the relevance part, constraints can be tailored towards specific exercise (types) and specific (structurally determined) configurations within a typical student solution. Additional requirements, which have to be fulfilled in that specific situation, are coded in the satisfaction part" (Menzel, 2006, p. 31).
- **Model-tracing** models maintain a model of problem solving that is 'traced' (compared) against a student's actions. "Feedback during problem-solving is given based on current state of the model (also called working memory) and the rules that represent student cognition and action" ("CTAT Basics," 2013). As an alternative to model-tracing tutors, *example-tracing* tutors "evaluate student behavior by flexibly comparing it against generalized examples of problem-step guidance on complex problems while recognizing multiple student strategies and maintaining multiple interpretations of student behavior" (Aleven et al., 2009, p. 105). These tutors are much less costly compared to model-tracing models because no detailed domain knowledge is required as the use of generalized behavioural examples suffices to build the model.

Although their assembly might be difficult and time-consuming, model-tracing student models allow the tutor to actively simulate the actions of the student, which is useful to predict future student behaviour.

Box 2.4 – Model-tracing student models are most informative for language tutoring.

2.2.3 Tutoring knowledge

A tutoring model has two main functions, which are mirrored in the basic tasks of instruction, namely to stimulate and evaluate learning. ITS research has mainly addressed these functions separately (VanLehn, 2007) and sometimes together as in the ASSISTment system (Razzaq, Patvarczki, et al., 2009; Razzaq & Heffernan, 2010; Turner, Macasek, Nuzzo-Jones, Heffernan, & Koedinger, 2005), which combines "assistance" and "assessment". A tutoring model thus needs to decide on *when* and *how* to intervene and it is responsible for content planning of what to teach next. The question of when and how to assist the learner is "the fundamental dilemma of tutoring, as discussed in the papers entitled *To Tutor or Not to Tutor: That is the Question* (Razzaq & Heffernan, 2009) and *Does Help Help?* (Beck, Chang, Mostow, & Corbett, 2008)" (Bourdeau & Grandbastien, 2010, p. 125).

A tutoring model decides on *when* and *how* to intervene and is responsible for content planning of what to teach next.

Box 2.5 – Tasks of a tutoring model

Assisting and tutoring the learner can further be divided into two sub-functions (Bourdeau & Grandbastien, 2010, p. 125): "cognitive diagnosis, defined as the detection of the sources of errors, and the selection of tutoring or remediation strategies". Recent developments in automatically learning the learner's affective states (Arroyo et al., 2009; Walonoski & Heffernan, 2006; Woolf, Arroyo, Cooper, Burleson, & Muldner, 2010) have increased the complexity of reasoning about optimal tutoring decisions.

A tutor's decisions are often reflected in the different forms of interaction that the tutor has with the learner. Typical forms of interaction include socratic dialogs, hints, feedback from the system, etc. A human teacher typically uses six types of feedback (Ferreira, Moore, & Mellish, 2007; Lyster & Ranta, 1997; Panova & Lyster, 2002): explicit correction, recasts, clarification requests, metalinguistic feedback, elicitation, repetition or any combination of these.

These interactions usually occur through the user interface module, that connects the student with the tutoring module (see Figure 2.4). The user interface often includes a dialogue system for interacting with the student. This type of conversational interaction is particularly useful when the learner's answer is incomplete. Because tutors usually have an approximate sense of what a student knows, it "appears to be sufficient to provide productive dialogue moves that lead to significant learning gains in the student" (Graesser, Chipman, Haynes, & Olney, 2005).

2.2.4 Constructing an ITS

Building a complete ITS from scratch is a huge task as the development of every component requires many decisions and large-scale software implementations. Three main approaches can be distinguished to build an ITS for a specific problem solving domain:

1. Shell-based techniques;
2. Authoring tools;
3. Software pattern languages.

Shell-based approaches are well-established in the field of Artificial Intelligence. They date back to the beginning of expert systems research. A shell can be defined as "a software development environment containing the basic components for building expert systems" (Nkambou, Bourdeau, & V., 2010, p. 362). The first shell-based approach was done with E-Mycin (Crawford, 1987), "a general purpose Expert System shell derived from Mycin", which was built to extend the inference mechanisms of Mycin to other domain knowledge. Shell-based approaches focus mainly on the system components and show little attention for the user interface. Some shells focus on curriculum planning, user modeling or content acquisition (Vivet, 1988; D. Sleeman, 1987). Others target all components (Goodkovsky et al., 1997), in a simple component implementation with "procedural models of the tutor's activity, tutoring criteria and constraints" (Park Woolf, 2008). Nkambou, Bourdeau, and V. (2010, p. 363) identify FITS as a good example of such a shell (Ikeda & Mizoguchi, 1994).

Secondly, authoring tools have become popular since their first appearance in the mid-1990s (including systems such as Dasher, Libra, MacLang, and winCALIS). Their success is mainly due to the high implementation costs (high time/product ratio) of shell-based approaches and the fact that they are not accessible to teachers. Therefore, authoring tools were launched to bridge this gap by taking care of many of the programming details and allowing "developers to focus more fully on instructional design and pedagogical procedures" (Fischer, 2013). However, even though authoring tools are easy to use

by nonprofessional developers (such as foreign language instructors), they "were also subject to substantial instructional constraints because their tools were designed for a specific purpose (e.g., to make written drill-and-practice exercises or multimedia listening comprehension lessons)" (id.).

A third approach to ITS building is to view intelligent tutoring systems themselves as software. Pattern language for ITS (PLITS) is a platform that provides developers with interesting patterns that helps them to build their own ITS (Salah & Zeid, 2009; Zeid & Salah, 2010). PLITS "was built from pattern mining by reverse-engineering many existing ITSs" and it was used to build the Arabic Tutor, a web-based ITS that teaches a subset of the Arabic language (Nkambou, Bourdeau, & V., 2010, p. 369).

Another alternative to build ITSs is to view them as multi-agent systems (MAS), which can be employed for building basic ITS infrastructures. Intelligent tutoring systems fulfill all the requirements to be viewed as MAS (Nkambou, Bourdeau, & V., 2010, p. 369), because: (i) they are made of different interconnected, complex components; (ii) they provide multiple, different and complementary services; (iii) each of their components is functionally autonomous; and (iv) they are equipped with specific knowledge structure and reasoning mechanisms. Examples of ITSs that use MAS techniques include ITSs to show a set of Agent-Oriented Software Engineering (AOSE) methods derived from ITS research (Vicari & Gluz, 2007). Similar to the shell-based approaches, agents are sometimes only used to build specific subcomponents of an ITS: the tutor (Mengelle & Frasson, 1996) or the learner (Vassileva, Mccalla, & Greer, 2003). Moreover, Nkambou, Bourdeau, and V. (2010, p. 369) also mention the use of MAS to "target a specific ITS service (e.g., planning, dialogue management, collaboration) or the whole system (Capuano, Marsella, & Salerno, 2000; Hospers, Kroezen, Nijholt, op den Akker, & Heylen, 2003; Nkambou & Kabanza, 2001).

Multi-agent systems can be employed for subcomponents of a tutoring system or for a complete system that is made of different interconnected, complex components.

Box 2.6 – Multi-agent systems are useful for creating an ITS

2.3 Examples of language tutors

Now that the history of ITS has been clarified and the general components of a system explained, it is time to turn to some real examples of language learning tutoring systems. Gamper and Knapp (2002) have made a full analysis of 40 existing ICALL research prototypes until 2002. They set up a classification framework that could analyze

ICALL systems according to five dimensions: the supported languages, the applied AI technologies, the language skills which can be trained, the language elements which can be learned, and the availability of the systems. However, their analysis revealed that the application of the proposed technologies is not yet mature: "[m]any interesting and promising systems remained in a prototypical stage" (Gamper & Knapp, 2002, p. 338). The examined systems were all research initiatives and did not yield commercial end products. Often systems were abandoned after the end of a project and many of them were never used in real language learning environment — often because of lack of funding.

The current section presents and evaluates example systems of more recent academic language tutors (Section 2.3.1) as well as three commercial language tutors (Section 2.3.2) that are still in use today. It highlights the main challenges that they are addressing and the drawbacks that these systems are faced with. The main goal of including these examples here is to be able to situate and compare their objectives and outcomes with the research prototype that is presented in this dissertation.

2.3.1 Academic language tutors

Research prototypes developed in universities tend to focus on some specific aspects of the ICALL enterprise: the diagnosis and analysis of errors, student modeling, the user interface, etc. In what follows I highlight three language tutoring systems that have been developed within academia, each concentrating on a different aspect. (a) The Canada-based *E-tutor* system for learning German specializes mainly on error diagnosis and correction. (b) The Portuguese tutoring system *TAGARELA* pays attention to automatically generating individualized feedback on spelling, morphological, syntactic and semantic errors. (c) The *ICICLE* system that assists native speakers of American Sign Language in practicing their English writing skills pays special attention to the student model (SLALOM, (Michaud & McCoy, 2001)), which tries to capture the status of the grammatical structures of English that the learner possesses.

German Tutor

The German Tutor was developed by Trude Heift and colleagues at the Simon Fraser University in Canada to assist students of German (from beginners to advanced learners) in training their language skills (Heift & Nicholson, 2001). The German Tutor allows students to build sentences with some indicated words and detects grammatical and other errors. Several language processing modules analyze the input (spelling, word usage, grammar, punctuation, etc.). The feedback components of the system correlate the detailed output of the linguistic analysis with an error-specific feedback message. "If a module detects an error, further processing is blocked until the student corrects the

The screenshot shows a web-based language tutor interface. At the top, it says "Guten Tag, John!" with a "Umlaute + ß" button. Below that, the instruction "Bilden Sie einen Satz mit den folgenden Wörtern." is displayed. A progress bar indicates "Übung 4 von 10" and lists the words "(def. Artikel) / Zeit / laufen.". A text input field contains the sentence "Der Zeit läuft.". To the right of the input field are buttons for "Prüfen", "Lösung", and "Weiter >>". Below the input field, the feedback message "Da ist ein Genusfehler bei dem Subjekt." is shown.

Figure 2.5 – An advanced learner makes a determiner-noun error and receives the following feedback: "the subject contains a gender error". Beginner learners would have received feedback that stated: "the determiner DER is not correct here" and intermediate learners are presented with the message "There is an agreement error between the determiner and the noun". This screenshot is taken from Heift and Schulze (2003b).

mistake" (Gamper & Knapp, 2002). The Student Model is based on student subject matter performance. It provides feedback and remediation that is suited to the learner expertise (Heift & Schulze, 2003b).

The German Tutor is a web-based tutor that can be accessed through www.e-tutor.sfu.ca. Students create a login for the system, which saves their learner model for future sessions. According to the information on its website, the etutor also provides error-specific and individualized feedback by performing a linguistic analysis of student input and adjusting feedback messages suited to learner expertise. This personal approach is possible thanks to the student model that the system makes use of. This student model can be used to identify the learner's strengths and weaknesses, information that is accessible by students. Moreover, the weaknesses can be used to provide exercises that focus on previous difficulties.

Three levels of proficiency are distinguished by the German Tutor: beginner, intermediate and advanced. They are distinguished through the skill-specific score for each grammar skill, with scores from 0-10 matching the beginner category, 11-20 intermediate and 21-30 advanced. Instructional feedback can then be modulated according to the error that was made and the skill level of the error type (see Figure 2.5).

TAGARELA

TAGARELA, which stands for "Teaching Aid for Grammatical Awareness, Recognition and Enhancement of Linguistic Abilities", is a web-based language tutoring system for learning Portuguese. It was created by Amaral and Meurers (2007), who describe their system as an intelligent electronic workbook: "[i]ts activity types are similar to the ones found in traditional workbooks, and are divided into six groups: reading, listening, description, rephrasing, fill in the blanks, and vocabulary". TAGARELA extends traditional

CHAPTER 2. ARTIFICIAL INTELLIGENCE IN EDUCATION

workbooks in the use of individualized feedback that is automatically generated.

The TAGARELA system (currently version 3.1) is now further developed by the Theoretical Computational Linguistics group at Tübingen University. It can be accessed through <http://sifnos.sfs.uni-tuebingen.de/tagarela/index.py/main>, once you own a login name and a password for the system. It was first designed to be fully integrated into the Portuguese individualized instruction program at the Ohio State University (Spring 2007). It is now also employed in Portuguese courses at UMASS Amherst.

The general TAGARELA architecture shown consists of six modules: Interface, Analysis Manager, Feedback Manager, Expert Module, Instruction Model, and Student Model. The analysis manager is the interface between the input sentence created by a student and the instruction and student models to decide on the best processing strategy. The system further "calls the appropriate sub-modules into the expert module to analyze the input" (Amaral & Meurers, 2008, p. 585). At the end of processing an annotated representation of the learner input is passed on to the feedback manager, which decides on the best feedback message to generate.

ICICLE

ICICLE (Michaud & McCoy, 2006) is a system developed to provide writing assistance to native speakers of American Sign Language (ASL) who are learning English as a second language (<http://ftp.udel.edu/research/icicle>). The system performs analysis on pieces of writing that are uploaded by users through a graphical user interface with several windows (see Figure 2.6). It then subsequently analyses the text and determines any grammatical errors with a range of text parsers and a CFG grammar consisting of 321 English language rules together with a number of mal-rules that represent commonly-committed grammatical errors (by deaf learners of English). Finally the system constructs a response in the form of tutorial feedback. The student can then resubmit his original text and the cycle starts over again.

The objective of its student model, called SLALOM (Michaud & McCoy, 2001), is to diagnose the current proficiency of particular grammatical structures of English in terms of "acquired", "being-acquired", and "unacquired". Grammatical concepts are grouped in a hierarchical fashion, based on studies in second language acquisition that looked at transfer effects in learning. SLALOM uses the hierarchy "to identify the current state of knowledge of a learner and to predict the next grammatical structures to be acquired" (Amaral & Meurers, 2007). Different from The German Tutor, it does not ignore the "usefulness of knowing that an individual may exhibit competence or preference for specific structures while he or she struggles on others" (Michaud & McCoy, 2006, p. 27), but makes use of a model of the *interlanguage* between ASL and English.

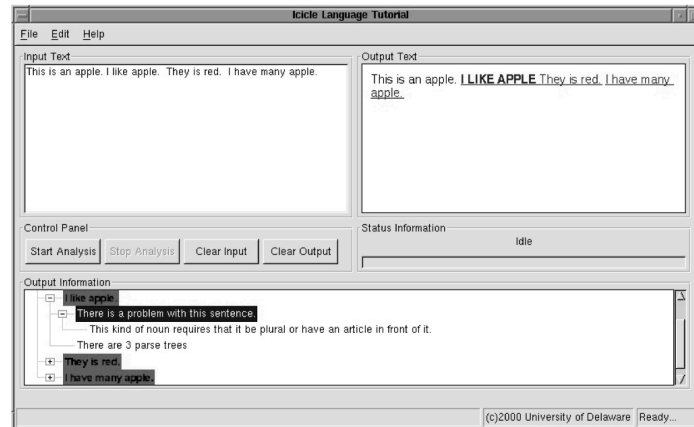


Figure 2.6 – The graphical application of ICICLE with several windows for the input text, output text and output information (below). This screenshot of the ICICLE implementation was taken from Michaud and Mccoy (2006).

2.3.2 Commercial language tutors

Language learning software has been around for more than 20 years in the form of digital exercises (fill the gap) that helped students practicing vocabulary words or specific parts of the grammar (such as tense, determiners, agreement, etc.). Although most of the systems are relatively mediocre in their goals and promises, there is one very popular system that stands out: *Rosetta Stone*. With bold claims about immersive learning environments, advanced automatic speech recognition and learning “twenty ways to say *I love you*”, they convince many learners to pay 400 euros to obtain one of their software licences. This section describes the main features of Rosetta Stone and two alternative commercial systems and discusses the main criticisms that scholars have formulated against it. The alternatives described below are the free web-based language learning system *Duolingo*, an initiative of Luis von Ahn in his attempt to translate the world wide web for free (von Ahn, 2013b), and the 2011 start-up company Games for Language that offers language tutorial video games online.

I selected these three systems because they each are representative of a particular business model. While the prestigious Rosetta Stone is available on CD-rom or with a 12 month access online, Duolingo is a web-based system that is available for free and access to the Games for Language web games is based on a one to three-month basis (for a democratic price). All three systems lack the notion of a student model and present all learners with exactly the same curriculum, only offering some shortcuts at certain points.

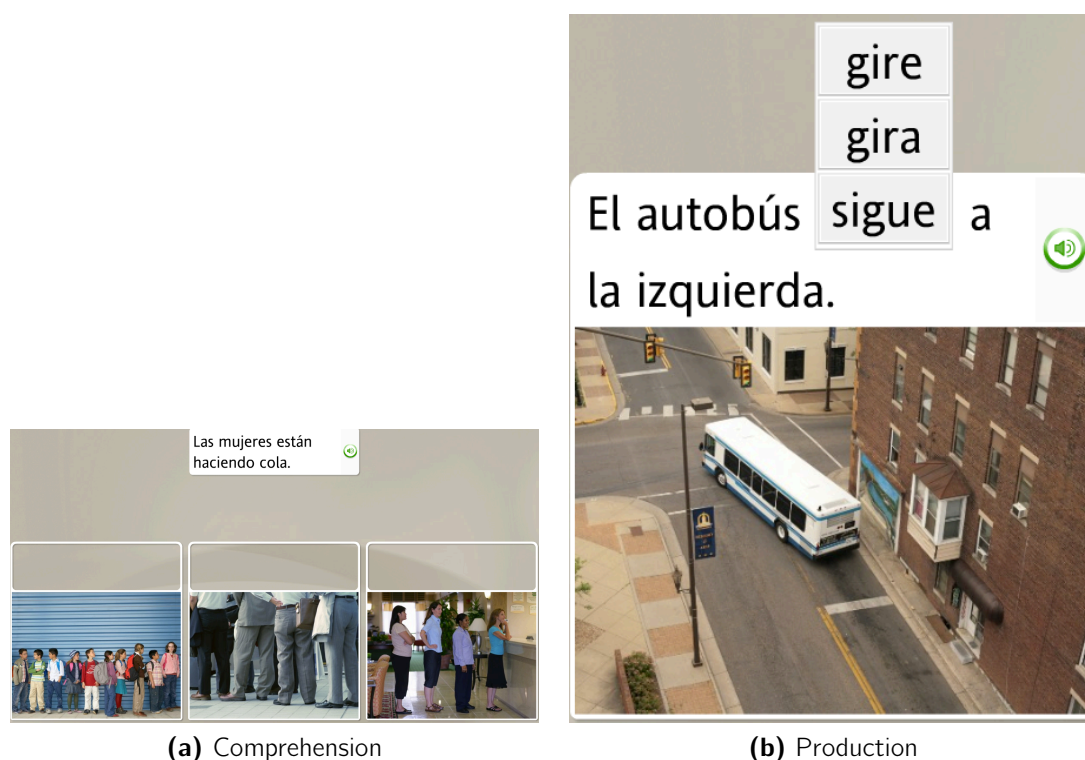


Figure 2.7 – Screen shots of an interaction with Rosetta Stone (Version 3), level 2, unit 1.

Rosetta Stone

Rosetta Stone is a commercial computer-assisted language learning (CALL) software package that uses images, sound, text and video to teach learners vocabulary and grammar by repetition at predefined intervals, without any translations. The main claim of the system is that it allows immersive learning of a language. By showing photographs to the student and asking him to select a photograph that a native speaker just describes (out of four options), to orally repeat what was said, or to complete a textual description of a photograph with multiple choice or an on-screen keyboard (see Figure 2.7), the student receives immediate feedback about his answers. Answers are scored on a scale from 0 to 100. Given that there are always four options to choose from, the student scores four marks for answering correctly on the first attempt, three for the second try, two for the third and one for the last option.

The courses come in three to five levels depending on the language and can be purchased separately. In February of 2013, they offered 25 Rosetta Stone language courses through their online website <http://www.therosettastone.com>. The software is used by the United States Army as a special military version of Arabic was offered to help troops in the Middle East to use phrases that are important in a military situation (Rosetta Stone, 2007).

However, despite its successes there are also many critical voices arising amongst language acquisition experts. The biggest one is probably the fact that - completely ignorant of the language you are learning - you will always be shown the same photographs and learn the same words depending on the level you are at. As Nielson (2011) points out, "The authors (of Rosetta Stone) claim that 'by combining genuine immersion teaching methods with interactive multimedia technology, Rosetta StoneTM replicates the environment in which learners naturally acquire new language' (pp. 2-3). This claim is clearly wrong. "The *Rosetta StoneTM* interface simply presents learners with matching activities in which they guess or use a process of elimination to determine which words or phrases go with particular pictures"(Krashen, 2013, p. 1).

The evaluations that have so far been carried out can be accessed through the Rosetta Stone website, but neither of them has been conducted by language learning specialists. The studies were run on adult subjects, mostly older than college students, that were learning Spanish. It is unclear who funded these evaluations. According to these reports, the participants in both studies were enthusiastic about Rosetta Stone: In the first study, Vesselinov (2009) "administered a questionnaire: 94% of the subjects agreed or strongly agreed that Rosetta Stone was easy to use, 88% agreed or strongly agreed that it was helpful and enjoyed using it, and 77% said they were satisfied with it" (Krashen, 2013, p. 2). Although the second study states that "perceptions of the Rosetta Stone solution were overwhelmingly positive" (), there is no data provided to support this claim.

Duolingo

Although Duolingo is available online for free, it has a clear business model behind it and can thus be seen as a commercial system. It has a clear goal in mind: translating the world wide web through the input of language learners. There are a range of investors that support Duolingo, such as NEA and Union Square Ventures.

A range of languages is supported by Duolingo: Spanish, English, French, German, Portuguese and Italian. In any of these languages, the learner is presented with translation exercises, with increasing levels of complexity as you proceed further on the so-called "skill tree". As a real translator, your task is to translate sentences from the foreign language into your first language (only English and Spanish are supported). You have the option to listen to the foreign sentences and when new words are introduced, you have the option of looking up their translation (see Figure 2.8a). For instance, a Spanish learner with English as a first language might be presented with the utterance *Yo soy una niña* that needs to be translated into *I am a girl*. If the word *niña* is still unknown to the learner he can look up its meanings by clicking on it.

Duolingo gives you the feeling that you are playing a game on your GameBoy while you are learning a language: you can lose hearts for incorrect answers, you practice

CHAPTER 2. ARTIFICIAL INTELLIGENCE IN EDUCATION

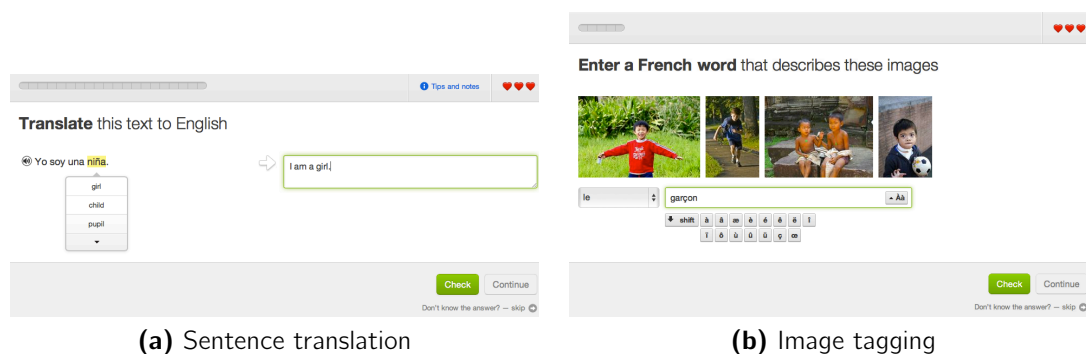


Figure 2.8 – Screen shots of an interaction with Duolingo, downloaded from the resources page on the Duolingo website (21 February 2013).

against the clock, you can level up depending on your progress, etc. Learning should become "addictive", according to their own slogan. The skill tree is organized in terms of themes such as food, animals, clothing, sports, etc. Some nodes in the tree focus more on grammar (e.g plurals, possessives, etc.), whereas other concentrate on vocabulary training. You earn a new medal at every level and you can track your progress inside one level in terms of the scores you obtained and the days of the week that you were interacting with the system (see Figure 2.8b for a screenshot of a user's skill tree).

The four components of language learning are addressed by Duolingo (von Ahn, 2013a, p. 2):

- Reading. The user is asked to translate from the object language into the native

language. The output of this exercise, correct or incorrect, will always make sense.

- Writing. The user is asked to translate from the native language into the object language. The user understands the input, but may produce nonsensical output. But combining these first two exercises over a number of users cooperatively produces output as good as professional translation.
- Listening. Rather than translate, to learn listening the user is asked to subtitle video in the object language. This again produces accurate output, augmented by automated processes such as spellchecking.
- Speaking. The user is asked to speak the object language in order to train a speech recognition algorithm.

A research study that tested the effectiveness of Duolingo was conducted in September-November of 2012 based on a random representative sample of native speakers of English learning Spanish. According to the report that describes the study, it was estimated "that a person with no knowledge of Spanish would need between 26 and 49 hours (or 34 hours on average) to cover the material for the first college semester of Spanish" (Vesselinov & Grego, 2012, p. 1). This result is based on the language test's cut-off point for the second college semester and the 95% Confidence Interval of the effectiveness measure. The effectiveness of the learning process is ascribed to the motivation of the participants, "with people studying for travel gaining the most and people studying for personal interest gaining the least" (id.). Also, it was found that the system was more effective for students with no initial level of knowledge of Spanish, compared to more advanced learners.

Games for Language

The start-up *Games for Language* is meant to be a tool to "boost your French, German, Spanish and Italian" with an online language learning course. You can run a demo version for free online for any one of these languages. This demo teaches you some basic expressions that are used on an airplane conversation between a passenger and an air stewardess. All exercises require you to click on the correct words when you see or hear their English translation. Some of them target agreement between an article and a noun (in a balloon game), others focus on particular pronouns (in a fishing game), etc.

Games for Language is a very playful way to practice your vocabulary in a certain language, especially when you are a beginner. You can score points by clicking on the correct balloons, cards, fish, clouds, etc. (see Figure 2.9). Yet, the results are not used by the program to steer future exercises. They are merely shown to the user so that he gets an understanding of his level of the target language, of course according to the pre-set exercises. Grammatical explanations are offered at certain moments in a game



Figure 2.9 – Vocabulary exercise that asks the learner to find the correct translation for the English word *apple*. All four words refer to fruit.

regardless of the correctness of the user's answer (e.g. the difference between *ser* (to be - permanent) and *estar* (to be - temporary)).

2.4 Conclusion

Although the first AI techniques started being used in the early 1970ies, the commercial systems that we find today still resemble the old systems of programmed instruction, where every student is taken through the same curriculum. Yet, the academic initiatives show more interesting language tutors with a full-blown ITS architecture and a more individualized task selection and feedback generation, although most of the running systems today are still far from the ideal of real personal tutors. The most common explanations for the general lack of success of full-blown CALL systems include, among others:

1. The **pedagogical principles** underlying most existing CALL systems stimulate extrinsic student motivation, based on scores and ranks, rather than intrinsic motivation.
2. Given that building a tutor is an enormous investment, there is **not enough funding** to support research-based tutoring systems for a long period of time.
3. The **programming skills** of good teachers that know how to engage their students are often not good enough to create intelligent tutoring systems themselves. The authoring tools are often too limiting to really be creative and manage the tutoring process in your own way.

2.4. CONCLUSION

The next chapter discusses my vision on effective pedagogical methods for language learning that can help to engage students in long-term interactions with the tutoring system. The lessons learned from the issues with certain state-of-the-art systems and the pedagogical ideas they support are incorporated into the fresh design of tutor agent in Chapter 9.

Part II

Operationalization

Chapter 3

Fluid Construction Grammar

The first prerequisite for any tutor, whether human or artificial, is full competence of the knowledge domain that is being learned by the student. Being instructed by a domain expert can really boost your motivation as a learner to achieve similar results in a natural way. If your ski teacher looks like a natural talent when coming down black slopes, you are likely to become motivated to practice more on your skiing technique to become just as good as him. Now, the same principle applies when constructing an artificial language tutor: he should first of all be a competent language agent that can produce and comprehend utterances in the target language and detect non-grammatical expressions. To build this agent, one would need to rely on a computational language formalism that can store all the linguistic knowledge that a language agent needs and process any utterance in the language.

Although there are a handful of computational grammar formalisms available today, such as Head-driven Phrase Structure Grammar (HPSG) (Pollard & Sag, 1994; Müller, 1996; Copestake & Flickinger, 2000), Sign-based Construction Grammar (SBCG) (Michaelis, 2009; Sag, 2012), Embodied Construction Grammar (ECG) (Bergen, 2003; Bergen & Chang, 2005) and some others, I have chosen to use the Fluid Construction Grammar (FCG) formalism here to operationalise the grammatical knowledge and procedures of the language agent. The FCG language processing framework has been under development since the late 1990s, when it was introduced to model language evolution and language change in agent-based simulations. However, thanks to its high degree of flexibility in learning new linguistic constructions and non-grammatical language processing, it can serve as a valid formalism for language tutoring purposes.

FCG allows its users to create grammars according to ideas that are found in Construction Grammar. Different from existing linguistic proposals for Construction Grammar (Goldberg, 1995, 2006; Croft, 2001; P. Kay & Fillmore, 1999; Michaelis & Lambrecht,

CHAPTER 3. FLUID CONSTRUCTION GRAMMAR

1996), FCG has a particular focus for processing issues that goes beyond the level of verbal descriptions of a language. Instead of committing itself to a specific implementation of grammatical features, FCG wants to be an open instrument that can be used by linguists to test out their theories or to set up language learning experiments that require flexible language use. The system is integrated with a web interface for visualizing the application of constructions during language processing and the resulting constructs. Its latest release (August 2013) can be downloaded as part of the Babel framework on ai.vub.ac.be/trac/babel2.

The FCG formalism has matured much over the last few years with new inventions, implementation optimizations and a standardization of the grammars and the use of FCG. Moreover, two volumes have been published recently that uncover the basic principles of FCG, provide formal reconstructions and a set of case studies that describe linguistic phenomena in areas ranging from German case, Hungarian verbal agreement, Polish negation, Russian aspect, to Spanish modality and Dutch posture verbs. Individual papers can be downloaded on www.fcg-net.org. The full references to these books are:

- Steels, L. (Ed.) (2011). *Design Patterns in Fluid Construction Grammar*. John Amsterdam/Philadelphia: John Benjamins.
- Steels, L. (Ed.) (2012). *Computational Issues in Fluid Construction Grammar*. Berlin/Heidelberg: Springer.

Like many other computational linguistics efforts, "the FCG-system is embedded within a contemporary Common LISP-based programming environment from which it inherits well-tested mechanisms for representing and processing complex symbolic structures" (Steels, De Beule, & Wellens, 2012, p. 197). As Fluid Construction Grammar is still partially under further development in our team, I am not a passive consumer of the formalism but actively work on finding new solutions for technical or linguistic issues on a daily basis. Specific extensions that were made to the formalism to realize the Spanish grammar needed in this dissertation are described in Chapter 4, Section 4.3.2.

This chapter describes the main features of the FCG formalism and explains how the language agent's construction inventory is built up and how a grammar engine usually functions. These are the two first components of a language agent's architecture. The third component, flexibility strategies, forms the focus of Chapter 5. To see the construction inventory and the grammar engine at work, you can explore the Spanish case study in Chapter 4.

3.1 The construction inventory

The construction inventory constitutes the grammatical knowledge of an agent. It is a collection of so-called "constructions" that are organized according to certain principles. The notion of construction is inspired by its use in Construction Grammar, a linguistic school of thought with a special focus on usage-based approaches to language (Goldberg, 1995, 2006; Croft, 2001; P. Kay & Fillmore, 1999; Michaelis & Lambrecht, 1996).

This section first introduces the notion of a construction and how constructions are represented in the FCG formalism (Section 3.1.1). Section 3.1.2 then discusses the organization of constructions in sets, inventories and networks. Finally, Section 3.1.3 explains how grammars can be constructed quickly, by using a range of design patterns that are available in FCG.

3.1.1 Constructions

A construction is the most basic data structure in FCG. The notion of a construction has been at the heart of linguistic theorizing for centuries (Fried & Östman, 2004). A construction is "a regular pattern of usage in a language, such as a word, a combination of words, an idiom or a syntactic pattern, which has a conventionalized meaning and function" ((Goldberg & Suttle, 2010), cited by Steels, De Beule, and Wellens (2012)). It is thought to capture both the usage pattern as well as the knowledge that a speaker has about the pattern.

Constructions relate meaning to form through a range of semantic and syntactic categorizations. There are specific constructions that relate meanings to their related semantic categories, others that map semantic categories into syntactic counterparts and constructions that express syntactic categorizations into a form (usually a word or a morpheme). These four connections are often referred to as the *grammar square* (see Figure 3.1). A good example of constructions that operate on all four axes of this square are argument structure constructions, as in the utterance *John kissed Mary*. John is captured by a lexical construction (meaning \leftrightarrow form) but his *kisser* role is translated into a more general semantic agent role by a construction that maps the *kisser* meaning into the semantic category for agent. The translation of agent into subject is handled by the active voice construction. Finally, because John is the subject of the sentence, word order constructions operate to put him at the beginning of the sentence.

The two ends of one arrow in the grammar square (e.g. meaning vs. form) make up two *poles* in an FCG construction. By default, FCG constructions link meaning (left pole) and form (right pole) or semantic categories and syntactic categories (the horizontal axes in

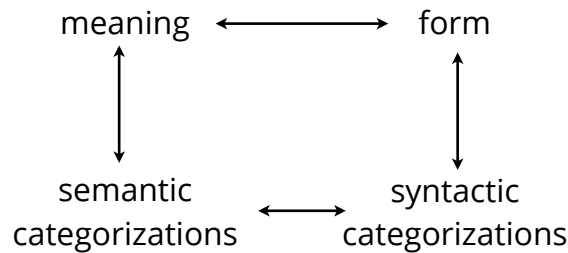


Figure 3.1 – Constructions operate on all four axes of the grammar square.

Figure 3.1). However, constructions consisting of two semantic poles or two syntactic poles can also be represented and processed. I will refer to these constructions as sem-sem or syn-syn constructions. Three example constructions (sem-syn) are included here below (taken from Steels (2011, pp. 5-6)):

1. Lemma's like "walk" are covered by *lexical constructions*. They express a direct association between a string (syntactic pole) and its meaning (semantic pole). "Lexical constructions also introduce additional syntactic and semantic categorizations that are important for later grammatical processing, such as the lexical category (part of speech), number or gender" [id.: p.5].
2. A *determiner-nominal construction* is a phrasal construction that creates a determiner-noun phrase out of a combination of a determiner (such as "the" or "some") and a noun (such as "table" or "mouse").
3. A *postposed-genitive construction* (e.g. "No son of mine") combines a nominal phrase with a preposition ("of") and a genitive ("mine"). The semantic pole adds the possessive relation between the referent of the nominal phrase and the referent of the genitive.

The remainder of this section discusses more technical details of constructions as they are implemented in FCG, including: their realization as a coupled feature structure, how hierarchy is dealt with and the special operators that work on their feature specifications. Readers with a basic knowledge of FCG constructions can skip this entire section.

Coupled feature structure

In technical terms, a construction is a *coupled feature structure*: it consists of two feature structures (one on each pole) that are coupled, which means that they are always used together. An FCG feature structure typically contains a list of units, which are usually mirrored on both poles. The units can be arranged in a tree structure (or any other structure) through special features that are responsible for building hierarchy, such as

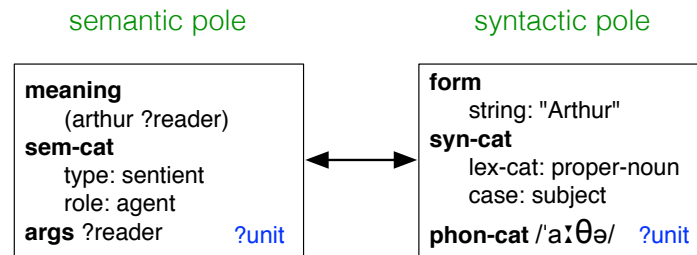


Figure 3.2 – A lexical construction tags information from that is present in the top unit and drags it into a newly created unit hanging under the top. The new unit also provides the features that characterize the lexical construction. J-units are special units that can create new units or modify existing ones (see 3.1.1). This figure is a schematic representation of the usual list structure.

sem-subunits and **syn-subunits**. A single unit consists of a list of features, which are attribute-value pairs that define the properties of the unit.

A construction typically contains values that are variables (e.g. ?var). These variables will be instantiated when the construction is used in a real sentence, where they get bound to the meaning or form values that are present in the situation. A schematic example of a construction (without subunits) is included in Figure 3.2, which depicts the lexical construction for the proper name Arthur. It has three semantic features: **meaning**, **sem-cat** (semantic category) and **args** (arguments). Its meaning is here represented in a straightforward first-order logic list notation, but depending on the use and the goals of the construction inventory, different meaning representations can be processed (e.g. procedural semantics or frame-based semantics). The meaning feature still contains a variable, which reoccurs here in the **args** feature: ?art. The arguments feature is used as an index by other constructions that build further on this lexical construction: e.g. the Proper-Noun Construction or the Intransitive Construction. On the syntactic pole, the features **form**, **syn-cat** and **phon-cat** are found. A form feature typically contains strings, morphemes or word order information. The syntactic category here also contains a feature with a variable ?case, which will be filled in by the appropriate argument structure constructions. Also a phonetic category is present, which can also be used by other constructions to determine the plural marker, a compound pattern, etc..

Constructions contribute to the formation of a *transient structure* or *transient linguistic structure*, which is also a coupled feature structure that is representing all the knowledge that a speaker or a hearer has of the utterance that is being processed. Figure 3.3 provides a detailed example of a transient structure for the utterance “Arthur read newspapers”. You see that the semantic and the syntactic poles are almost mirror images except for unit 4, which is only present on the syntactic pole. This unit contains exclusively form features that express the plural form of “newspaper”.

When the transient structure is finished (stopping criteria are discussed below) the parsed

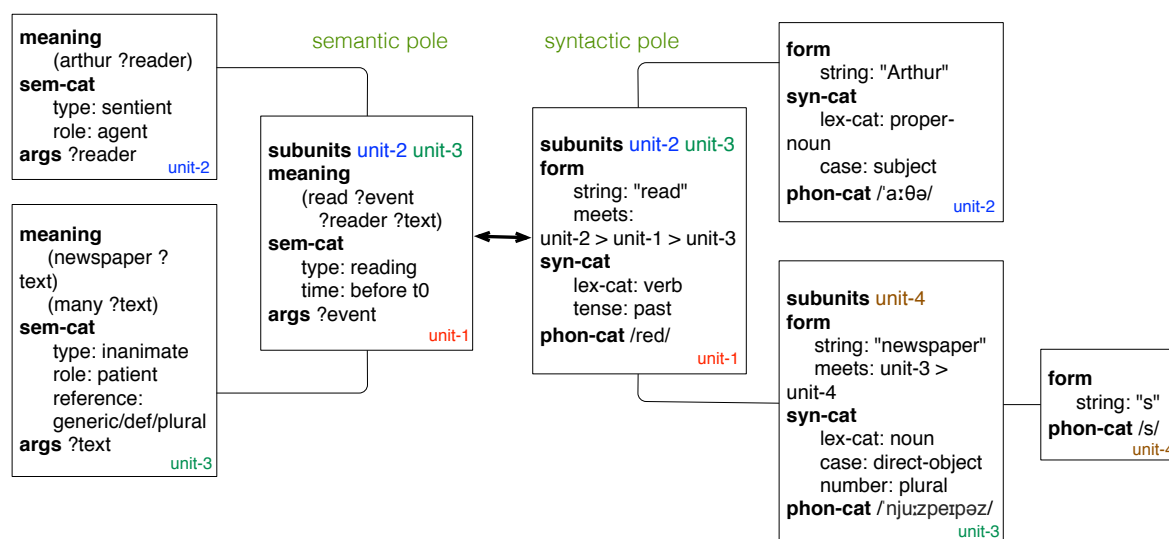


Figure 3.3 – A coupled feature structure consists of a semantic (left) and a syntactic (right) pole, each with several (mostly symmetrical) units. The coupled feature structure in this figure is the result of parsing/producing "Arthur read newspapers".

meaning or produced utterance is extracted from it through its **meaning** and **form** features. For instance, the resulting meaning of the parsing process of "Arthur read newspapers" is collected into one list: ((read ?event ?reader ?text) (arthur ?reader) (newspaper ?text) (many ?text)). The variables indicate that the arguments are open and can be filled according to the current context of the utterance. The importance is that they are shared across multiple meaning predicates. For instance, ?reader occurs in the read predicate and also in the arthur predicate. This means that a phrasal construction related these variables through the **args** feature, so that the arthur meaning is now the agent of the event.

The following sections provide more details on some of the special features that are present in the feature lists of a construction, including J-units, tags, footprints and special operators. All these features are relevant for processing, which is fully discussed in Section 3.2.

J-units

The multiple levels and relations that occur between units of a feature structure (see Figure 3.3) are created by a special type of units that are used in a construction specification: J-units. J-units are instantiated by the *J-operator*, which defines operations on an existing or non-existing unit. To distinguish it clearly from other units a J-unit specification does not start with the unit-name (e.g. ?top) but instead with a list starting with the symbol J (e.g. (J ...)). A J-unit enumerates operations for a single unit,

3.1. THE CONSTRUCTION INVENTORY

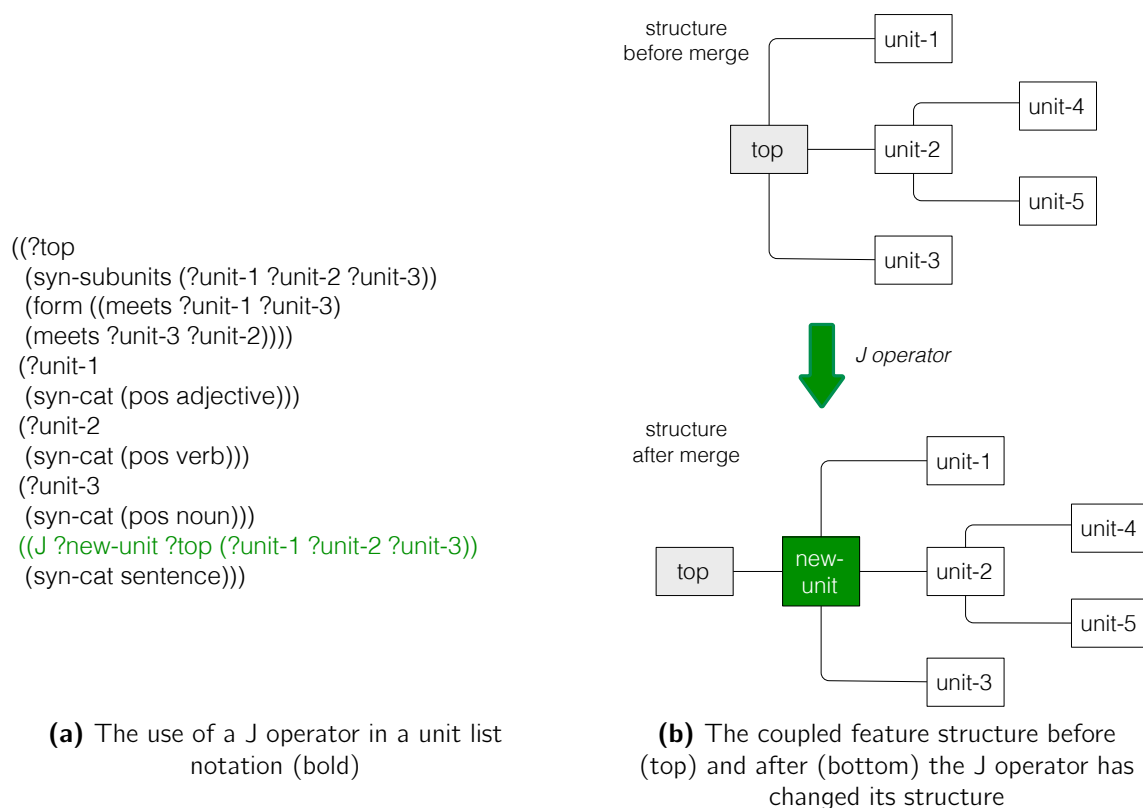


Figure 3.4 – The J operator creates hierarchy in a transient structure.

linked through a reference to its unit name immediately after the J in the list call. Of course a feature structure can contain multiple J-units, thereby allowing operations on multiple units. An example (syntactic) feature structure with a single J unit looks as follows:

```

((?top
 (form ((string ?top "big"))))
 ((J ?new-unit)
  (syn-cat ((pos adjective)))))

```

The above feature structure consists of a single unit `?top` and a J-unit. It will create a new unit (`?new-unit`) containing the syntactic category adjective. The body of a J-unit (i.e. the part after the initial list) resembles that of a regular unit in that it can contain feature value pairs. However, the newly created unit does not have any specific location yet in the feature structure tree. To do this, the J-operator has two optional parameter to specify a unit's parent unit and optionally its daughter units. If you want to hang the new unit under the existing top unit, you can do this as follows: `(J ?new-unit ?top)`. Daughter units can be specified by adding a list as the last parameter of the J-operator call (see Figure 3.4).

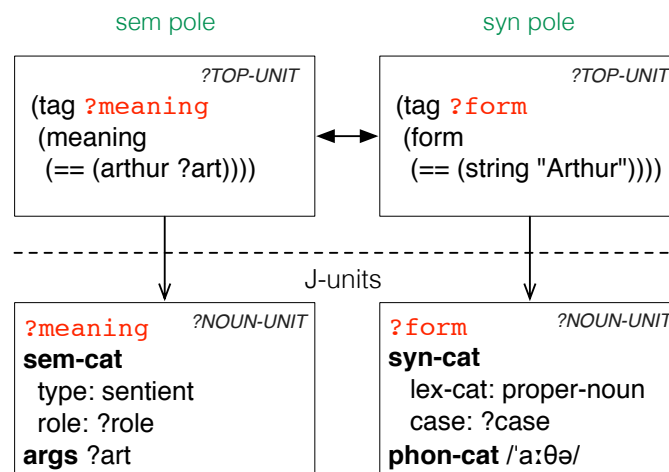


Figure 3.5 – Tag features move features between units. The complete feature that follows the red tag name (**?meaning** and **?form**) will be moved to the newly created **?noun-unit** after this construction could apply.

Tags

A tag is a special feature that allows to move certain features inside one pole of a coupled feature structure (see Figure 3.2). Its most wide-spread use is in lexical constructions, where meaning or form features are moved from the top unit of the transient structure into a newly created lexical unit. The syntax of tag features is defined as follows:

(tag ?tag-name feature-that-you-want-to-move)

When the tag name (*?tag-name*) is repeated inside another unit of the same pole, the feature(s) that follows it will be deleted from the original location and added to the new unit. Tag features are frequently used inside the J-units (see Figure 3.5 for an illustration of the tag feature use in a lexical construction). Apart from moving features from the top, tags are also used to remove a particular feature permanently from a feature structure - an operation which is usually not without further implications. This removal operation can be carried out by repeating the tag name inside a unit that is named **nil**. Nil units are ignored by the transient structure, so any feature that is moved here disappears from the transient structure.

Footprints

A footprint is usually added to an existing feature structure during construction application. It overtly signals the application of a construction and thereby prevents a construction to apply twice on the same structure. Footprints are just like tags optional features that one can use to build feature structures in FCG.

Usually the `footprints` feature contains the construction name of the construction that added it to the transient structure: e.g. `(footprints (arthur-cxn))`. But also other information can be added to its feature value list such as the construction set (e.g. `lexical`; see Section 3.1.2) or the type of the construction (e.g. `irregular verb`). This additional information can then be used as conditional checks that allow other constructions to apply. A regular conjugation construction could for instance only apply in production after there was no irregular construction that could express the conceptualized meaning. To indicate whether a particular footprint is expected or is prohibited in a particular unit special operators can be used that formulate set operations.

Special Operators

A coupled feature structure usually contains more than only units and their respective feature specifications. Special operators were introduced to allow for a richer representation inside units that facilitates set operations. FCG lists can contain special operators that indicate how the elements of a list are treated in unification. The operations that are currently supported by FCG are listed in Table 3.1.

Table 3.1 – Special unification operators in FCG.

operator	description
<code>==</code>	includes: the elements of the list can occur in any order
<code>==1</code>	includes uniquely: the elements of the list can occur in any order and there are no duplicates allowed
<code>==0</code>	excludes: the elements following are disallowed
<code>==p</code>	permutation: the elements of the list can occur in any order and no additional elements are allowed

When used in a construction, special operators are the first element in a list of feature values (see Figure 3.6). For instance, the `==` operator is often used in the `meaning` and `form` features to indicate that there can be more meaning predicates or form strings than the ones named here and that their order is irrelevant to the application of a construction. The use of the `==1` operator in the `syn-cat` and `sem-cat` features implies that the values of these features cannot contain doubles, e.g. there is only one semantic type or one lexical category allowed for a single unit here. The excludes operator `==0` is often used to disallow the presence of certain footprints in a particular unit, especially when these footprint features are added by the construction in another unit.

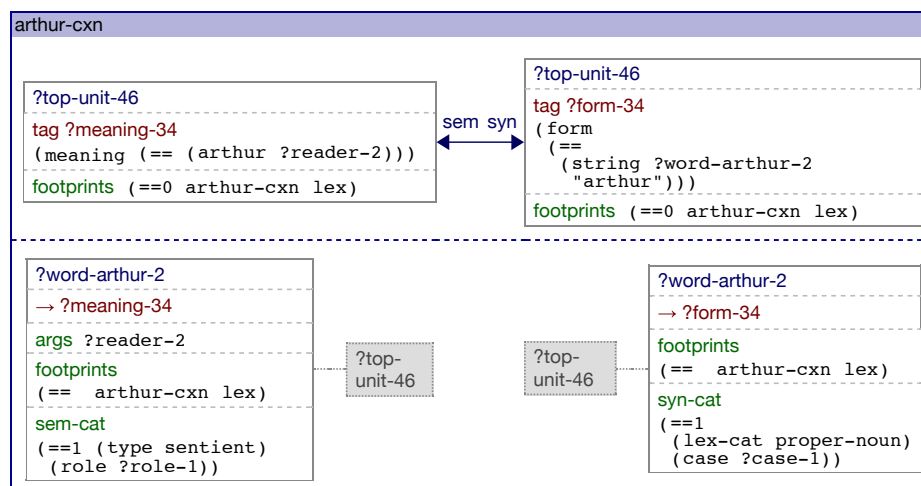


Figure 3.6 – This screen shot of the FCG web interface shows the use of special operators (`==`, `==0` and `==1`) inside the Arthur Construction.

3.1.2 Organizing constructions

Rather than being independent entities, constructions typically form an ordered collection that is organized according to some predefined principles. This ordered collection is referred to as a *construction inventory* in FCG. A construction inventory is a data structure that can be configured depending on the language that it is used for, the linguistic competence that the language agent has or the personal preferences of the grammar engineer.

An FCG-construction inventory is typically organized in three ways: (i) having no explicit organization, (ii) in construction sets or (iii) in a network of constructions. By default, there is no explicit organization imposed on the constructions. However, by means of heuristics that can be added to the search process and the inherent conditional nature of construction application in FCG, the inventory organisation is not completely blind. This section further describes constructional organization in terms of construction sets and construction networks.

Construction sets

Constructions can be grouped into sets according to the nature of the work that they carry out (Croft & Cruse, 2004). In this way, lexical constructions can be grouped into a set as they all express single words in isolation, or phrasal constructions can be joined in a set, given their shared grouping behavior. A set organization allows for an explicit order between the construction sets, so that one set can be considered before the next one. In FCG, this order typically differs in parsing and production. Frequent construction set orders are, in parsing: `lexical` → `morphological` → `functional` →

grammatical, and in production: lexical → functional → grammatical → morphological.

Construction sets have proven to be useful both from a processing point of view, and from an implementation point of view:

- In terms of processing, the search process can be constrained considerably (see Section 3.2.2). Because the FCG-interpreter needs to go "through all constructions in the inventory in order to find one that could usefully apply, and if there is more than one, a branching of the search path must be organized" (Beuls, 2011, p. 238). When construction sets are used, only constructions belonging to a particular set need to be considered when searching for the next construction.
- In terms of an effective implementation, construction sets allow you to get a better overall structure of the grammar so that the grammar design can focus on different construction set. This approach helps to cope with the inevitable complexity of working out real grammars (Steels & Wellens, 2006).

Also inside a construction set, further subdivisions might be needed. A subset approach is often useful when a language employs unmarked defaults for certain expressions. For instance, when singular is not expressed overtly, the singular morpheme construction (which adds a zero morpheme \emptyset) should be applied after the plural morpheme construction was tried out. Unmarked constructions in a set can therefore be put in a separate subset of the morphological, phrasal or any other construction set.

A network of constructions

In a network organization, one construction can prime or take precedence over others. Apart from defining networks of constructions based on relations between individual constructions, the FCG-engine can also use this information in processing (Wellens, 2011; Wellens & De Beule, 2010). The key idea is that a chain of construction applications implies the existence of dependency relations between constructions. Therefore, linking the constructions through these dependencies gives rise to an intricate network of constructions and categories since it turns out that it is grammatical categories that most often constitute these dependencies. Not only can such a network help in disambiguation but it can also be used to vastly speed up future processing by prioritizing or priming constructions whose dependencies have been met.

One of the case studies on which this the network priming idea has been tested on is the Hungarian verb agreement system. This system shows an intricate example of agreement that alternates between subject-verb and object-verb agreement depending on deictic information that is present in the utterance (Beuls, 2011; Beuls & Wellens,

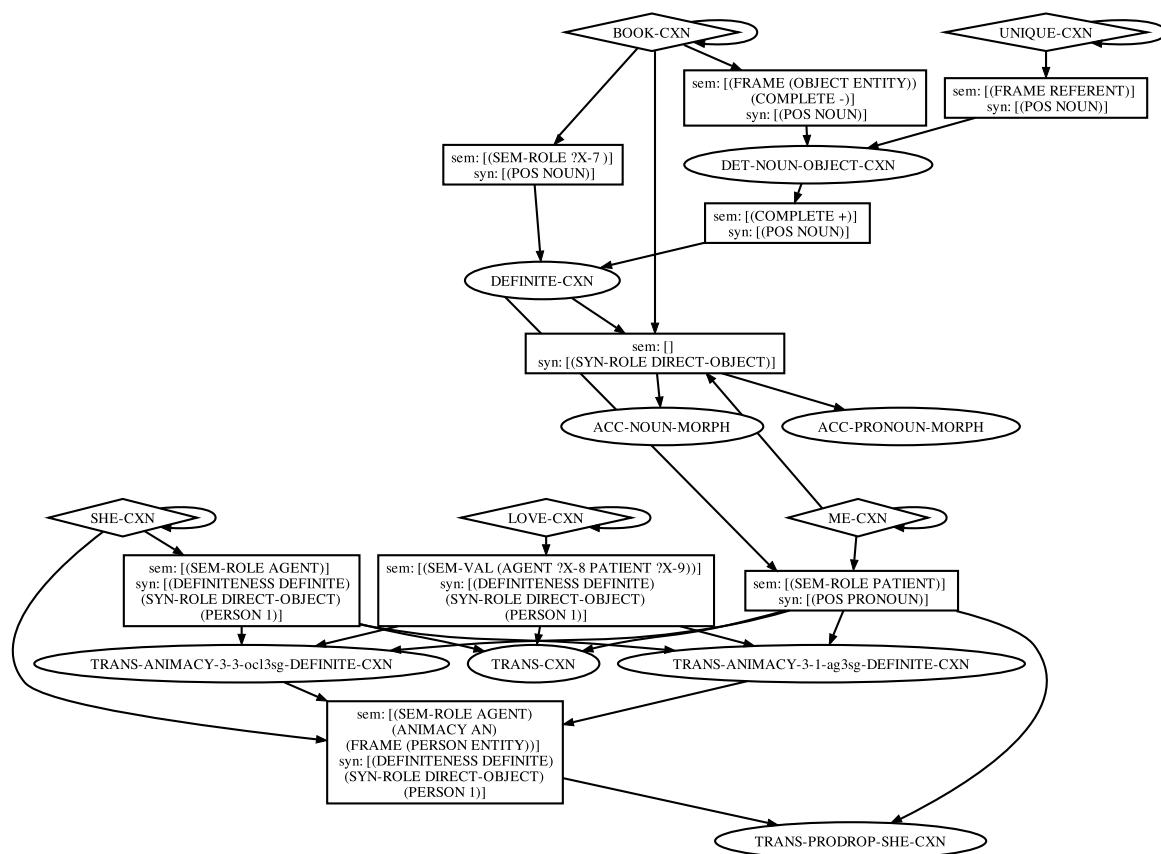


Figure 3.7 – The dependency network as learned after processing sentences 1 and 2. Diamond shaped nodes represent constructions that could be applied independently of any other construction (i.e. on the initial linguistic structure). Egg-shaped nodes represent constructions as well but those that were observed to have a dependency with other constructions. Square nodes hold semantic and syntactic categories that essentially constitute the dependencies.

2010). The alternation makes this a good case study for the dependency networks. The following examples have been learned by the dependency networks (see Figure 3.7):

- (1) Szeret-i a könyv-et
love-3sgDEF the book-ACC3sg
He/she loves the book.
- (2) Szeret-∅ engem
love-3sgINDEF me-ACC1sg
He/she loves me.

Through the dependency network some constructions are primed over others. In this way, the accusative construction that adds the *-et* ending to the book stem can be primed as soon as the argument structure construction did its work and set the case category of

the book unit to accusative. Priming occurs thus through specific categories (features) inside constructions as relations between features are learned over time. The network in Figure 3.7 is dynamic and will be updated as more and more utterances have been processed.

3.1.3 Templates

Templates make the necessary abstractions that allow the grammar engineer to quickly create large-scale grammars. Moreover, every individual template integrates one specific design pattern in language, such as agreement systems, field topology for syntactic structuring or a linking pattern for semantic structuring (Steels, 2012b). In sum, a template specifies certain aspects of a construction and one construction is built by several templates. The FCG templates that exist hide many processing features such as the footprints, tags, J-units, etc. In this way you can fully concentrate on the grammar you are developing and the linguistic features that are needed to do this. The volume on design patterns in FCG (Steels, 2011) contains many examples of FCG templates. I have included two illustrative examples below.

The first example concerns the lexical template `def-lex-cxn`. This template builds a lexical construction, such as the construction for the proper noun Arthur. A lexical template usually consists of two parts, linked together through the construction name, in this case `arthur-cxn`. The construction name can be freely chosen by the grammar designer. The first part (`def-lex-skeleton`) contains some of the key features of the lexical construction such as its form (`string`), meaning and arguments. The second part of the lexical template adds the categories to the lexical construction: `def-lex-cat`. It assigns semantic, syntactic or phonetic categories, some of which can still be undefined: e.g. a semantic role (`?role`) can only be filled in after the argument structure constructions have done their job.

```
(def-lex-cxn arthur-cxn
  (def-lex-skeleton arthur-cxn
    :meaning (== (arthur ?reader))
    :args ?reader
    :string "arthur")
  (def-lex-cat arthur-cxn
    :sem-cat (==1 (type sentient)
                  (role ?role))
    :syn-cat (==1 (lex-cat proper-noun)
                  (case ?case))))
```

Another example of a template that is used abundantly in this dissertation is the

`def-morph-cxn` template, which creates morphological constructions (see also (Gerasy-mova, 2012) for a more detailed explanation on this topic). In the example below, I included the morpheme for the third person singular present in English (“-s”). Because morphological constructions do not express semantic meaning but map syntactic function into form and reversely, they only contain `syn-cat` and `form` features:

```
(def-morph-cxn 3sg-present
  :morph "-s"
  :word-order meets
  :syn-cat (==1 (person/number 3sg)
            (tam (==1 (tense present))))))
```

Apart from the agreement information (`person/number`), the tense-aspect-mood (TAM) feature is crucial here for the realization of the verbal “-s” morpheme. The construction will apply to verb stems that meet these properties and do not yet have any other morphemes following the stem (see `word-order` feature). Chapter 4 contains many more examples of FCG templates that are specific to the goals of this thesis.

3.2 The grammar engine

FCG’s grammar engine is responsible for the linguistic processing of constructions, either in production or parsing. During this process, constructions build up a coupled feature structure, commonly called the *transient linguistic structure* (cf. Figure 3.3), which contains all the semantic and syntactic information that a speaker has about an utterance. To build a transient structure, two processes are required: (i) a search process to find the next construction that can expand the transient structure and (ii) a mechanism to unify the linguistic information in a construction with that present in the transient linguistic structure. This section treats both processes in Section 3.2.1 ‘Match and merge’ and in Section 3.2.2 the search process is explained in further details.

3.2.1 Match and merge

A transient linguistic structure can be extended through a process of so-called “construction application”. Construction application consists of two main phases: a *match* phase and a *merge* phase. Due to this separation in two phases, FCG differs hugely from other grammar formalisms that apply constructions “in one go” and call this process *unification*. In FCG terms, when the term unification is used, it exclusively refers to the matching process. Match and merge work in a similar fashion to if-then rules. The matching

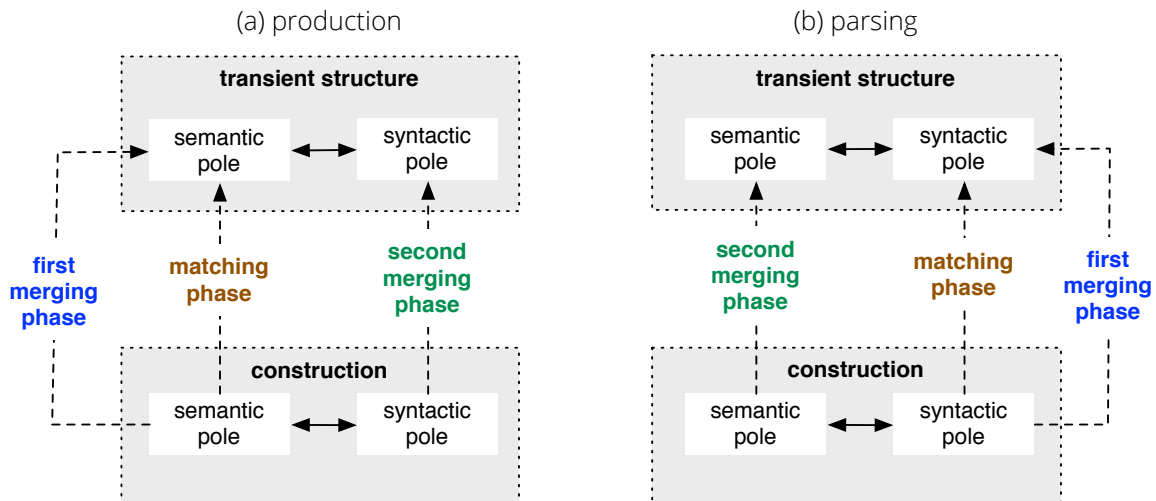


Figure 3.8 – Construction application in production (a) and parsing (b) consists of two main phases: matching and merging (with a first and a second merge phase).

phase checks whether certain conditions for construction application are fulfilled and the merge phase merges new information into the transient structure, potentially with more restrictions as to whether this information can be added.

Construction application is a directional process. Because an FCG construction, and therefore any coupled feature structure in FCG, consists of two poles, it plays an important role which pole is used for matching (the *if*) and which one for merging (the *then*). The direction depends only on the type of linguistic process: production or parsing. In production, matching happens on the semantic pole, which checks whether the semantic features needed by the construction are fulfilled (see Figure 3.8, right). Merging is split up into two parts: a first merging phase and a second merging phase. This split is inherent to the architecture of FCG with its strict separation of features in a semantic and a syntactic pole. The first merging phase merges new features into the matching pole (semantic in production). The first merging phase always relies on the information that is provided by the J-operator. The second merging phase, adds all the features that are provided by the opposite pole (syntactic in production), given that this does not lead to any conflicts in the transient linguistic structure.

If matching or merging (in any of its two phases) fails, the transient linguistic structure will not be extended with the particular construction that caused this failure. Instead, we retract the new information and try with the next construction that succeeds the matching phase. The process of finding new constructions to apply is what is called construction search in FCG. The next section is dedicated to search.

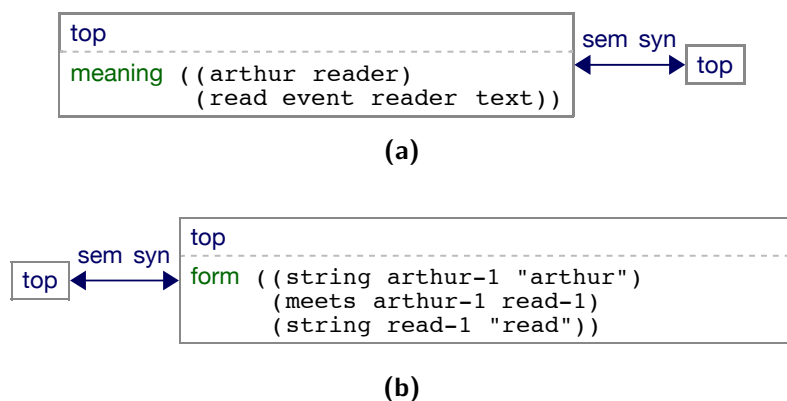


Figure 3.9 – The top-units of the transient structure at the start of production (a) and parsing (b).

3.2.2 Searching constructions

The initial feature structure from which the search process is started consists of two top units: one that is empty and another one that contains either a meaning feature (in production) or a form feature (in parsing) (see Figure 3.9). The top units are created by processes which run independently from the FCG engine, such as *conceptualization* or *de-rendering*. The latter is the process whereby an utterance is segmented before its form feature is created (typically consisting of *string* and *meets* features).

When more than one construction can extend the transient linguistic structure or one construction that can be applied in multiple ways, a split occurs in the search tree (see Figure 3.10). By default, "the search engine employs a *depth-first* approach that continues to explore a single hypothesis until no other constructions can apply anymore" (Bleys, Stadler, & De Beule, 2011, p. 149). By means of goal tests it is checked whether the resulting transient linguistic structure is preferable and, when this is not the case, the goal test can instruct "the processor to backtrack to an intermediary transient structure to which other constructions apply, possibly leading to a desirable final transient structure" (id.). If all possible hypotheses have been explored and still no desirable transient structure has been reached, the entire application process fails.

Apart from goal tests that are a stopping criterion for the search engine, the search process can be steered in the following ways:

- The use of *construction scores* can be taken into account so that constructions with higher scores are considered first by the search engine. Construction scores are typically updated after they have been used in successful communication.
- *Construction sets* (see Section 3.1.2) constrain the search space because only a subset of the full construction inventory needs to be searched through to find the

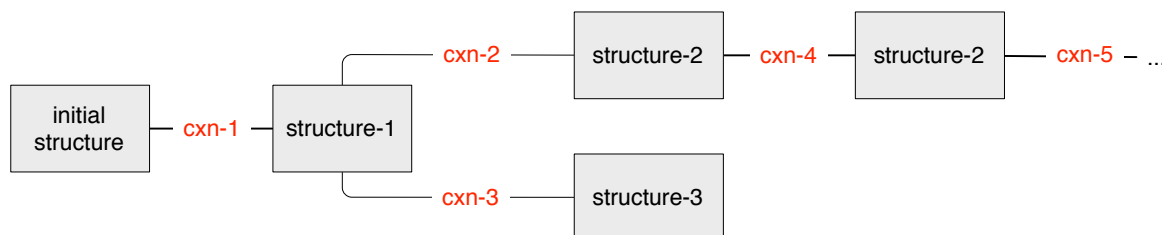


Figure 3.10 – Language processing is organized as a search process in FCG. A linguistic structure is being built up by applying a series of constructions to it. When two (conflicting) constructions could apply to the same linguistic structure, this leads to a split in the search tree.

next construction. When no further constructions of a particular set can apply, the search engine proceeds to the next set. The explicit order of construction sets is provided in the configuration of the construction inventory.

- Because the lexicon usually makes up the majority of all constructions present in a construction inventory, it can become very expensive to start searching for the next lexical construction as on average 50% of the full set will have to be tried out in matching before the desired construction applies. Therefore, working with a *hashed lexical construction set* can be more efficient. Instead of matching the construction on the transient linguistic structure, the meanings to be produced or the strings to be parsed can first be looked up in a hash table. In this way, the desired constructions can be prioritized in construction application.
- *Node tests* perform checks every time a new construction could apply, instead of only at the end of construction applications as the goal tests do. A default node test that is provided in FCG is *check-duplicates*. This node test verifies whether there are two feature structures in the search tree that have exactly the same structure, which can happen due to a different order of construction application in two branches.

All four search optimizations explained here are used in the grammars presented in this dissertation.

3.3 Conclusion

The main aim of this chapter was to introduce and support my choice for Fluid Construction Grammar (FCG) as a useful grammar formalism as a basis for a language tutoring architecture. The main design features of FCG can be summarized in the following four points:

CHAPTER 3. FLUID CONSTRUCTION GRAMMAR

1. A construction is a typical usage pattern in language. The knowledge that a typical speaker has of a certain construction is stored in the FCG constructional data structure, referred to as *coupled feature structure*. A *construction inventory* is a collection of constructions.
2. Because the same constructions are used in production and parsing, a construction has two *poles*, one semantic pole that is consulted in production when meaning needs to be expressed and one syntactic pole for parsing that contains form information and typical syntactic information.
3. To analyse or synthesize utterances the construction inventory is called upon by the construction application process, in which new constructions are *matched* with one pole to an intermediate parsing/production solution and if a match was established, the other pole's information is *merged* into the solution as well. Special operators are used to influence the matching and the merging operations by modifying the conditions on the feature structure unification.
4. *Construction application can be optimized* by a range of techniques such as goal tests, node tests, construction scores and (hashed) construction sets. But first and foremost, the design of constructions and the linguistic features that are used in them is the most important means of cost-efficient construction application.

Depending on your needs as a grammar engineer, the Fluid Construction Grammar formalism gives you the freedom to explore your own choices when you manually create a grammar. The linguistic features that you include in your constructions, the organization of a construction inventory and even the application process of constructions can be tuned to adapt to your particular situation. The high degree of freedom that the FCG formalism gives to the grammar engineer is perhaps one of the most important motivations to use it as a NLP component in an intelligent tutoring system for language learning. Moreover, FCG has already been used for many case studies over the last years (especially during the European FP7 project *ALEAR*; www.alear.eu). These case studies have led to the creation of more complex FCG grammars, including grammars for: German spatial language (Spranger & Loetzsch, 2011), Spanish pronominal clitics (van Trijp, 2010), Polish negation (Höfer, 2012), Russian aspect (Gerasymova, 2012), German determiners (van Trijp, 2011), Spanish modals (Beuls, 2012), Hungarian verbal agreement (Beuls, 2011), German information structure (Micelli, 2012) and English quantifiers (Pauw & Hilferty, 2012). This expansion of grammars and the knowledge that it has brought along was one of the triggers that led to the realization of the Spanish grammar for the tutoring architecture that is presented in this dissertation.

Yet, grammars are dynamic entities, especially when they are being constructed during language acquisition. Therefore, in a learning situation, constructions are never stable but can be extended or features can be modified after a particular learning event has occurred.

FCG, with the adjective "Fluid" in its name, has been developed with especially this open-ended type of communication in mind. The formalism provides hooks to implement problem-based learning that does not interrupt the routine construction application layer in a visible manner but instead calls on a meta-architecture to handle diagnosed problems. This aspect plays a crucial role in the remainder of this dissertation. Both the language agent and the student agent rely on this meta-architecture to diagnose and correct learner errors on the one hand (flexibility strategies) and the acquire new constructions or new usage variants on the other (learning strategies). Flexibility strategies form the topic of Chapter 5, learning strategies are treated in Chapter 7 when the student agent architecture is being dismantled. But before we turn to flexibility issues, the next chapter shows how to create an operational grammar for verb conjugation in Spanish.

Chapter 4

Spanish verbs in FCG

Spanish verb morphology is notorious for its complexity with up to 140 different conjugated forms for a single lemma (Bosque & Demonte, 1999) and – depending on the reference grammar you use – 89 different conjugation schemes. On the one hand a learner thus has to memorize according to which schema a particular verb lemma is conjugated and on the other hand the verbal endings that belong to that schema need to be recalled when producing a particular verb form. Moreover, even the most advanced learners of Spanish tend to struggle with the correct use of verb tenses, especially in the selection of the past tense (past perfect/imperfect) and the correct use of the indicative or subjunctive mood (Vanden Bulke, 2005; Wood Bowden, Gelfand, Sanz, & Ullman, 2010). These *formal and semantic challenges* make Spanish an interesting test case for a first implementation of a language agent. This chapter describes the construction inventory and the grammar engine components of the agent; the flexibility strategies are treated in the next two chapters.

Spanish verb conjugation poses a range of formal and semantic challenges that make it an interesting test case for a first FCG-based language tutoring architecture.

Box 4.1 – Motivation of the selected language system.

Before we turn to real examples of FCG constructions that are crucial for operationalizing Spanish verbs, Section 4.1 describes the main features of Spanish verbs, with a focus on their rich morpho-phonology and the semantic intricacies they bring along. Spanish-speaking readers can immediately skip to the FCG-specific implementation in Sections 4.2 (individual construction design) and 4.3 (construction application). This implementation is rather technical to allow other users of FCG to reimplement this grammar or to implement a grammar for the same language system in a different language. Finally,

Section 4.4 puts the current implementation into a larger perspective by comparing it with related attempts in formalizing inflectional morphology.

4.1 A complex grammar

A verbal suffix (also called verb ending) that follows a Spanish verb stem conveys information about the following five linguistic features: person, number, tense, mood and aspect. In Spanish, similar to Modern Greek, Italian or many other languages in the world, subject pronouns are not expressed overtly, except when the speaker wants to stress the subject. Therefore, the necessary information to identify the subject is present in the verbal suffix. For instance, the difference between the English *I speak* and *you speak* is in Spanish only discernible in the verb's inflection, as you can see in the following examples:

(3) Habl-o con Pablo
 speak-IND.PRS.1SG with Pablo
 I speak with Pablo

(4) Habl-a-s con Pablo.
 speak-IND.PRS.2SG with Pablo
 You speak with Pablo

Yet, although Spanish verb endings are naturally expressive because they contain agreement information, there are a few attested cases of syncretism in the Spanish verbal paradigm. In these cases, a single suffix can have more than one function, which means that either:

- The intended function becomes clear when the complete verb form is taken into account: the past imperfect *ía* ending becomes a conditional marker when it is combined with the verb's infinitive rather than its stem: *cantaría* 'he/she would sing' vs. *comía* 'he/she ate'; or
- The context in which the suffix is embedded reveals its function: *hablaba* can mean both 'I spoke' or 'he/she spoke' but in a discourse context it will become clear whether the speaker is talking about himself or a third person.

Due to the high amount of grammatical information that is present in the verb suffixes and the rich morpho-phonology of the Spanish language, a single verb lexeme can potentially have up to 140 different verb forms when its full conjugational paradigm is taken into account: 19 tenses/moods, seven inflected forms per tense/mood, two infinitives,

Table 4.1 – There are 19 tenses/moods for one inflected verb form. This table shows the 1sg conjugation of the verb *ayudar* 'to help' (Rello & Basterrechea, 2011, p. 229).

Tense/mood	Verb form (1st person singular)
present tense/indicative	<i>ayudo</i>
present tense/subjunctive	<i>ayude</i>
preterite imperfect tense/indicative	<i>ayudaba</i> (2nd person singular)
preterite imperfect tense/subjunctive 1	<i>ayudara</i>
preterite imperfect tense/subjunctive 2	<i>ayudase</i>
preterite perfect composed tense/indicative	<i>he ayudado</i>
preterite perfect composed tense/subjunctive	<i>haya ayudado</i>
past perfect tense/indicative	<i>ayudé</i>
past perfect composed tense/subjunctive	<i>hube ayudado</i>
preterite pluscuamperfect tense/indicative	<i>había ayudado</i>
preterite pluscuamperfect tense/subjunctive 1	<i>hubiera ayudado</i>
preterite pluscuamperfect tense/subjunctive 2	<i>hubiese ayudado</i>
future tense/indicative	<i>ayudaré</i>
future tense/subjunctive	<i>ayudare</i>
future perfect tense/indicative	<i>habré ayudado</i>
future perfect tense/subjunctive	<i>hubiere ayudado</i>
conditional simple tense/indicative	<i>ayudaría</i>
conditional perfect tense/indicative	<i>habría ayudado</i>

two gerunds and four participle forms (Bosque & Demonte, 1999) (Table 4.1). Additionally, a verb stem can have a variety of stem realisations, e.g. the verb *tener* 'to have' has four stems along with the default *ten-* stem: *tien-* (diphthongization), *teng-* (velar insertion), *tuv-* (irregular) and *tend* (assimilation with following future tense *-r*). A basic Spanish learner grammar identifies not less than 89 different conjugation schemes to cover the conjugation of all Spanish verbs (Mateo, 1998).

4.1.1 Formal complexity

To model the degree of formal complexity present in Spanish verb conjugation, it is useful to split the verb form into three parts: a stem (with optionally attached affixes), a tense/aspect/mood (henceforth: TAM) ending and a person/number (or agreement) ending (Figure 4.1). Also inside the stem itself a further segmentation is necessary to allow morphophonemic changes that lead to different stem realizations. The segmentation that is used in the presented FCG grammar consists of three parts: an onset, a nucleus and a coda. The nucleus contains the stem's main vowel: e.g. the *o* in *enc-o-ntr-* (< *encontrar*, to find). This information is saved in the lexical construction of the verb, more specifically as part of the phon-cat feature on the construction's syntactic pole.

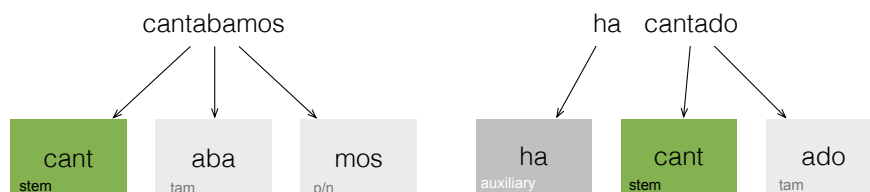


Figure 4.1 – A verb form such as *cantábamos*, we sang, is segmented into three blocks: (1) a stem *cant*, (2) a tense-aspect-mood (or TAM) ending *aba* and (3) a person-number (or p/n) ending *mos*. Note that word stress is not taken into account here. Each of these blocks, can be exchanged to create a new verb form: *cant-aba-s* 'you sang', *cant-a-mos* 'we sing', etc. Periphrastic forms are segmented as two separate words. Since irregular forms such as the auxiliary *haber* 'to have' are saved as lexical items in the grammar, *ha* 'he/she has' is not segmented any further.

Furthermore, verbs are divided into three verb classes: (i) *a*-themed verbs, (ii) *e*-themed verbs and (iii) *i*-themed verbs. The theme vowel is typically reflected in the TAM segment of a conjugated verb form. Verbs of classes 2 and 3 share most verb endings and only differ in certain fields of the conjugational paradigm. Examples of these three classes are *cantar* 'to sing', *comer* 'to eat' and *vivir* 'to live'. Furthermore, endings of first verb class verbs' indicative conjugations reappear in the subjunctive paradigms of verbs that belong to the second or third verb class: e.g. *cant-a* 'he sings.Ind' → *com-a* 'he eats.Subj'.

The stem-internal segmentation can be used to classify verbs into *verb types*, which define the morpho-phonemic changes that a verb can undergo. Typical changes that occur in Spanish verbs include:

1. **Stem vocalic nucleus changes** such as diphthongization ($e \rightarrow ie$; $o \rightarrow ue$) and raising ($e \rightarrow i$; $o \rightarrow u$);
2. **Stem post-nucleus changes** that guarantee a stable stem pronunciation: *coj-o* (< **cog-o* from *coger*, 'to take'), *juegu-e* (< **jueg-e* from *jugar*, 'to play'), etc.;
3. **Velar insertion** between stem and suffix: *ten-g-o* (< **ten-o* from *tener*, 'to have'), *val-g-o* (< **val-o* from *valer*, 'to merit'), etc.; and
4. **Assimilation** between stem and suffix: *le-yeron* (< *le-ieron*).

Although these changes can partially be predicted, they also often require additional knowledge about a verb's nominal counterparts in the lexicon: e.g. *jugar* ~ *un juego* 'a game' but *un jugador* 'a player'. Additional changes that occur include the use of diacritic marks on the last vowel of the stem to indicate a change in pronunciation: *actuar* 'to act' → *actúo* or *variar* 'to vary' → *varío*. Yet, similar verbs ending on *-uar/-iar* do not receive this diacritic mark on their conjugated verb forms and are pronounced as diphthongs:

evacuo 'I evacuate' or *odio* 'I hate'. According to the Real Academia Española, there is no general grammar rule that decides when to use diacritic marks on a stem's last vowel or when to keep using a diphthong (www.rae.es).

4.1.2 Semantic complexity

Time and tense are concepts that have fascinated linguists for a long time. I follow a standard formal account here that represents Spanish tense on two time axes (Bull, 1965, pp. 149-171): the present time sphere and the past time sphere. Bull (1965) has distinguished between systemic and non-systemic meaning. According to Whitley (2002, p. 110), Spanish and English concur in their use of systemic meanings that is "the meaning each category has within the overall system". Yet, in the case of non-systemic meaning the languages diverge more in the specialised functions the tense system has acquired by way of extension. I follow a systemic account here that tries to use a unified description for each tense category.

In Bull's model all situations are directly or indirectly related to the present point ('now'), which has also been referred to as a temporal zero-point (t_0) by other scholars (Declerck, 1991). The time of utterance always functions as t_0 . The tense system is then divided into two time-spheres: the past time-sphere and the present time-sphere. The past time-sphere is situated completely before t_0 . The present time-sphere includes t_0 and is divided by it into three parts:

1. **The pre-present sector:** the part of the present time-sphere lying before t_0 ;
2. **The present sector:** the part of the present time-sphere centered around t_0 ;
3. **The post-present sector:** the part of the present time-sphere following t_0 .

Figure 4.2 illustrates this model for the conjugation of the Spanish verb *amar* 'to love'. A distinction should be made here between absolute and relative tenses (depicted by the red crosses lying on or below the time axis). An absolute tense relates its situation directly to the t_0 . There are four absolute tenses in Spanish: the past tense (preterite), the present perfect, the present tense and the future tense. These tenses lie on the time line (crosses on the arrow in Figure 4.2). Tenses that are used to relate a situation to another situation are relative tenses. The figure also includes examples of conjugated verb forms for each of the depicted tenses.

Aspect is not explicitly represented on this time line, nor is mood. Although, both categories deserve an extended discussion on their own, they do not form the main focus of this dissertation. Therefore, the only difference between the two aspectual forms

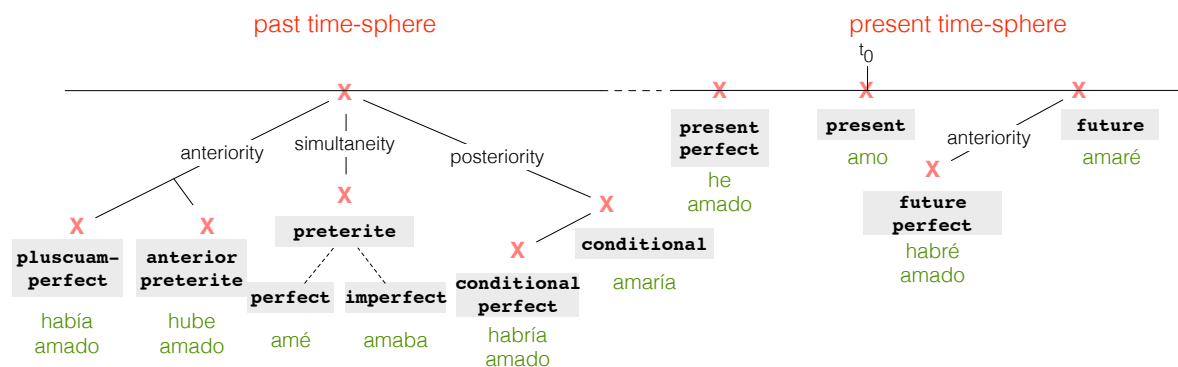


Figure 4.2 – Spanish has two main time-spheres: the past time-sphere (a) and the present time-sphere (b). The past time-sphere contains six tenses, related to a moment in the past through relations of anteriority, simultaneity or posteriority. The present time sphere contains four tenses, situated around the moment of speaking (t_0)

- The first singular verb forms for *amar* 'to love' are included to illustrate the morphology that accompanies the semantic distinctions.

that overlap the preterite time point in the past time sphere is the perspective that they include: internal (*in the middle of...*) for the imperfective and external (*ended or began*) for the perfective. Mood is not depicted on the time line in Figure 4.2 and I only consider the subjunctive present form in this dissertation, which is represented through a binding statement that binds the event to a different time axis, parallel with the present time sphere (see Section 4.2). For a more elaborate implementation of Spanish modals in FCG, previous work can be consulted that treats modal verbs and value judgements with indicative/subjunctive mood changes (Beuls, 2012).

4.2 A Spanish verb construction inventory

A language agent's construction inventory is defined as the collection of all constructions that he knows. Yet, it is also an ideal version of a competent language user's grammar, containing a complete description of the necessary constructions and their role in processing. This section describes the constructions that are needed to operationalize Spanish verbs in FCG. The construction inventory for Spanish verbs contains the following four types of constructions:

1. lexical constructions: constructions such as the *hablar-cxn*, a construction for the verb 'to speak', or the irregular construction *soy-cxn* for 'I am';
2. phrasal constructions for tense-aspect-mood and agreement, for example the *present-tense-cxn* (a construction that creates a present tense verb phrase)

or **intransitive-cxn** (the argument structure construction for the intransitive clause);

3. morphological constructions e.g. **aba-cxn** (a morphological construction for the TAM ending *-aba*); and
4. phonological constructions: e.g. **e→ie-cxn** (a construction for the diphthongization of a vocalic stem nucleus *e* into *ie*).

These different types of constructions are distributed broadly across the two constructional poles. Some of them are strictly operational between form and meaning poles or between semantic categories and syntactic categories, whereas others, such as phrasal constructions, connect meaning directly to syntactic categories (TAM features or case features). Also morpho-phonological constructions work only on the syntactic pole (connecting syntactic categories (and their functions) to formal expressions). The following sections include examples for every constructional type.

4.2.1 Lexical constructions

Lexical constructions are perhaps the most prototypical construction type because they establish a mapping between a meaning (the lexical meaning of the verb) and a form (the verb stem, infinitive, an irregular form, etc.). A lexical construction also receives some basic semantic and syntactic categorizations (Figure 4.3). Such a categorization consists of a list of feature specifications, such as a semantic class of event, a lexical category of verb or its verbal paradigm. The verbal paradigm is expressed through a feature matrix representation, to avoid too many overlapping constructions because many verbal endings are identical for the 2nd and 3rd paradigms (for more on the use of feature matrices in FCG see the introductory paper by van Trijp (2011)). For instance, a feature matrix for the *ía* ending (past imperfect 2nd/3rd verb class) indicates that the verb can either belong to the 2nd or to the 3rd verb class: (**verb-class** (==1 (1 -) (2+3 + ?2nd ?3rd))).

In the grammar that is presented here a construction's meaning is represented by a logic-based representation that is based on predicates and arguments: e.g. (**hablar** ?event-1951 ?context-1320). *Hablar* 'to speak' is thus a predicate corresponding to an event (?event-1951) that is situated within a particular context (?context-1320). Its arguments are also found in the **args** feature on the semantic pole. Because they are variables (identifiable by the question mark), the verb can be used for any particular speaking event in any specific context.

The form feature of a lexical construction contains the verb's stem in the form of a string

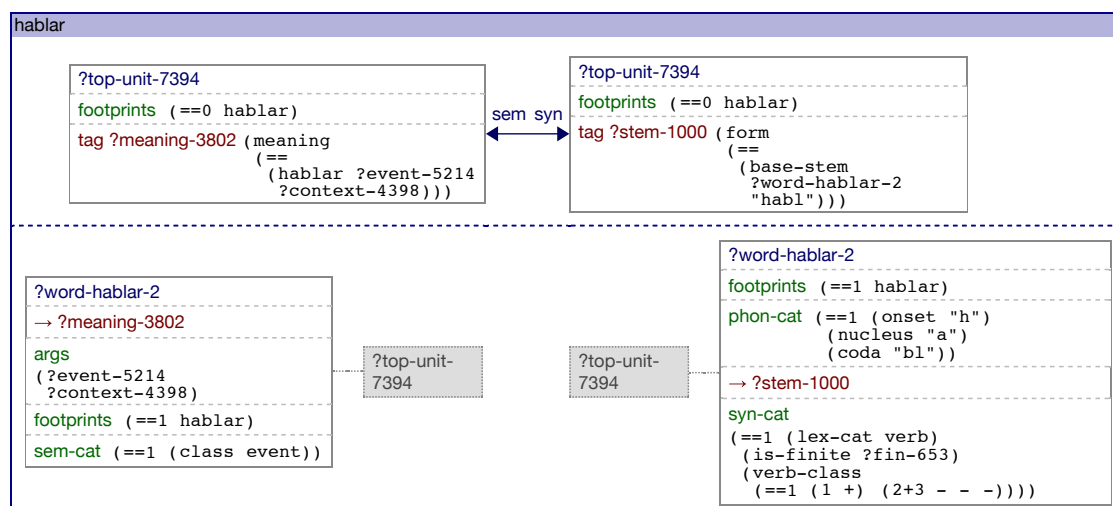


Figure 4.3 – This screenshot illustrates the representation of the lexical hablar-cxn for the verb to speak (stem "habl-") with a semantic pole (left) and a syntactic pole (right). The units above the dotted line represent the input that matches with the existing linguistic structure in parsing (syn-pole) or production (sem-pole). The lower units contain information that is added to the linguistic structure when this construction can apply. The lexical construction has a category (main verb) and contains information on the conjugation paradigm (1st verb class).

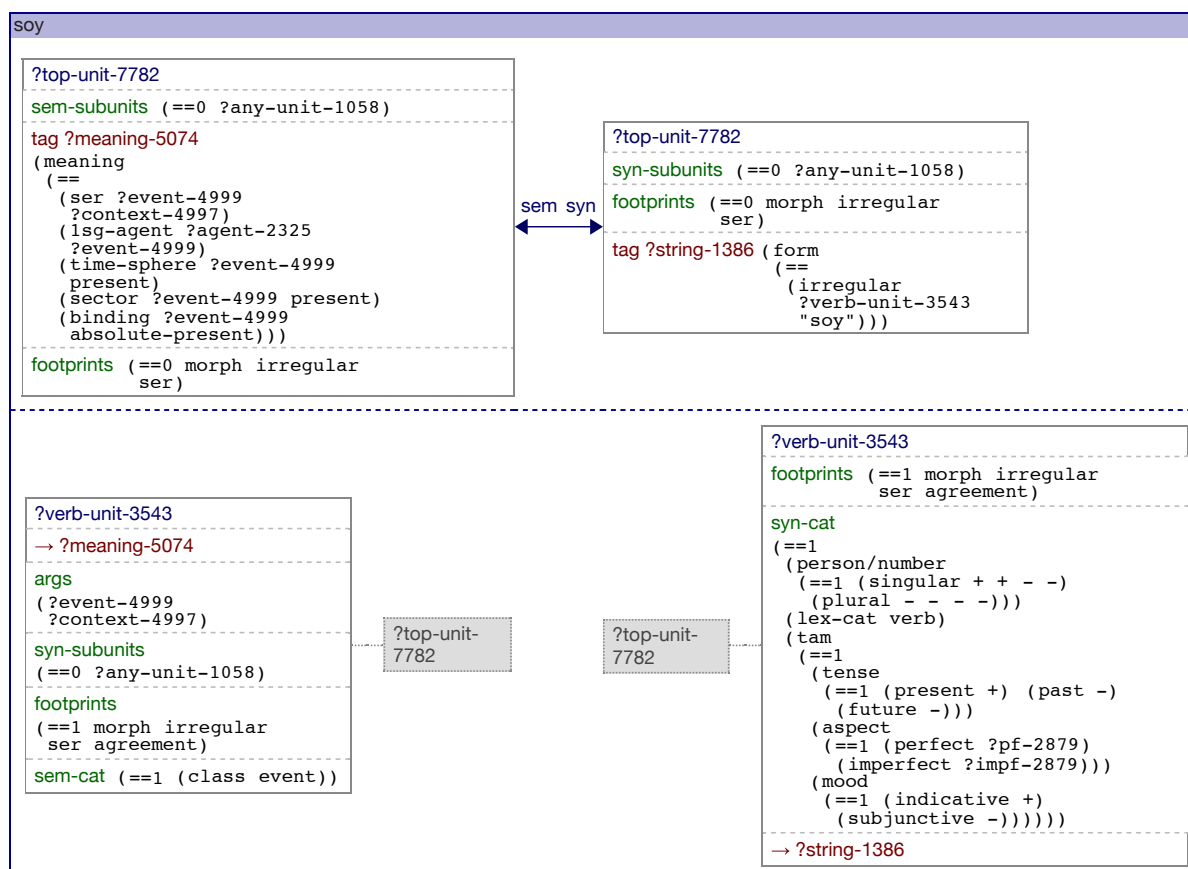


Figure 4.4 – The irregular lexical construction for soy maps the "soy" string straight onto its complete meaning. Also the syn-cat is already completely filled in.

such as "habl" for *hablar* 'to speak'. By adding the verb's stem to the form feature, it is thus assumed that the verb form is segmented before FCG processing is started, that is before the top unit is created in parsing. The pre-processing step that is required to do this segmentation in FCG is explained in Section 4.3.2. The verb stem is also further analyzed by the lexical construction according to the onset-nucleus-coda pattern, which is represented by the *phon-cat* feature on a construction's syntactic pole (Figure 4.3).

Also irregular constructions are captured by the lexical construction template. Because they are accessed as a whole by a language user and not segmented according to the Spanish verb segmentation rules, their form is mapped directly to the full meaning it expresses. For example, *soy* 'I am' maps to:

```
((ser event context)
 (1sg-agent agent event)
 (time-sphere event present)
 (sector event present)
 (binding event absolute-present))
```

In line with the semantic schema explained above (see Figure 4.2), the meaning of a single verb form consists of three to four components, on top of the lexical meaning of the verb itself: a time-sphere, a binding, a domain and (for aspectual meanings) a viewpoint. The subjunctive mood is captured through the binding meaning predicate. Some examples of tense-aspect-mood meanings for verb forms of *amar* are included here below. You can check the location of these forms on the time axis in Figure 4.2.

- *había amado*: time-sphere = past; binding = relative past; domain = anteriority;
- *hube amado*: time-sphere = past; binding = relative past; domain = immediate anteriority;
- *amé*: time-sphere = past, binding = absolute past; domain = simultaneity; viewpoint = external;
- *amaba*: time-sphere = past; binding = absolute past; domain = simultaneity; viewpoint = internal;
- *habría amado*: time-sphere = past; binding= relative conditional; domain = posteriority;
- *amaría*: time-sphere = past; binding= relative past; domain = posteriority;
- *he amado*: time-sphere = present; binding = absolute present perfect; sector = pre-present;

CHAPTER 4. SPANISH VERBS IN FCG

- *amo*: time-sphere = present; binding = absolute present; sector = present;
- *amaré*: time-sphere = present; binding = absolute future; sector = post-present;
- *habré amado*: time-sphere = present; binding = relative future; sector = post-present;
- *ame*: time-sphere= present; binding = subjunctive; sector = present.

Clearly, this semantic classification is just an example of one possible ontology for the representation of tense in Spanish. Aspect is merely represented by one additional predicate in the past tense: internal (*in the middle of*) or external. Also mood is one additional predicate that locates an event on an additional timeline, parallel to the regular one (**binding**). Of course, depending on your needs as a grammar engineer, you can include more complex meaning structures into the ontology that you feed to the grammar you design.

4.2.2 Phrasal constructions

Phrasal constructions combine separate units into phrases. The Spanish verb grammar currently contains two subtypes of phrasal constructions: constructions related to tense, aspect and mood (or TAM constructions) and agreement constructions that establish subject-verb agreement. They are language-specific and have thus been optimized for Spanish here. In their function of gluing units together, they often also map meaning predicates into a form feature and reversely. The form feature is here often an indication of the word order, or how to organize the units that are involved when the construction is processed. Occasionally, it happens that a phrasal construction creates a new phrase out of a single unit only. This is the case for some of the TAM constructions that cover non-periphrastic verb forms such as *habló* 'he/she talked': they create a full verb phrase out of a single verb unit (Figure 4.5).

An example of such a single subunit phrasal construction is the TAM construction for the past perfect tense (Figure 4.5). As its semantic contribution is concerned, this construction situates a given event (captured by the ?**verb-unit** variable) in a particular time sphere and specifies its relation to a reference point in that time sphere. On the syntactic pole, the newly created unit ?**verbal-phrase** receives the same syntactic information (: **syn-cat**) as the verb unit, completed with a **phrase-type** feature set to **verbal-phrase**. The creation of a verb phrase unit is beneficial in further processing, because the verbal information can be accessed as a whole by agreement constructions.

Agreement constructions apply immediately after the TAM constructions and create an

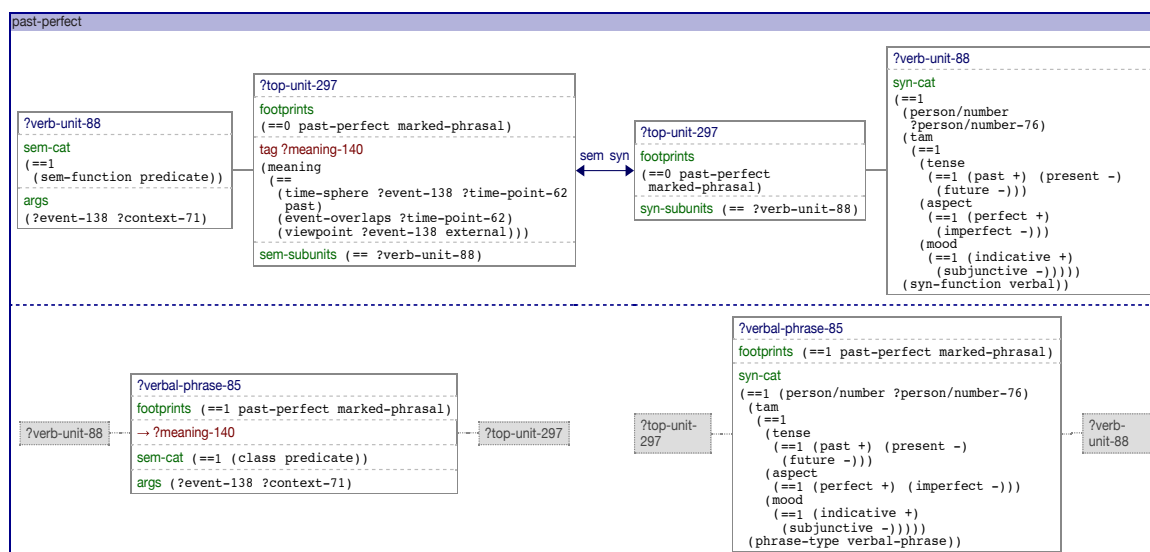


Figure 4.5 – The phrasal construction for the past perfect adds information on tense, aspect and mood to the verb form that is being created/analyzed. A verb phrase is created (phrasal construction) that can be accessed in further processing, such that the same agreement constructions can be used for single and periphrastic forms.

intransitive clause (subject only) that consists of the subject unit (if applicable) and the verb phrase unit. Because in Spanish the grammatical subject of a verb is by default expressed through the verbal endings, the proposed grammar consists of agreement constructions that express an additional agent meaning, such as (`1sg-agent ?agent ?event`) that stands for a 1st singular agent in an intransitive event. There are six agreement constructions present in the grammar: 1sg, 2sg, 3sg, 1pl, 2pl and 3pl.

4.2.3 Morphological constructions

Morphological constructions (or *morph-constructions*) are responsible for the most diverging expressions of verbal endings, which can convey an event's location on the time line, its agreement pattern or reveal its aspectual or modal meanings. They consist of two syntactic poles instead of a semantic and a syntactic one. Their main task is to map a certain syntactic categorization (or the construction's function) onto the corresponding surface-form and back. For a detailed discussion of the design and the function of morphological constructions, the interested reader is referred to the volume on *Computational Issues in FCG*, Chapter 5 (Gerasymova, 2012).

An example of a morph-construction is the -s ending that expresses subject-verb agreement for the 2nd person singular (Figure 4.6). It contains processing conditions that apply in production and others that are used in parsing. The three conditions that guide

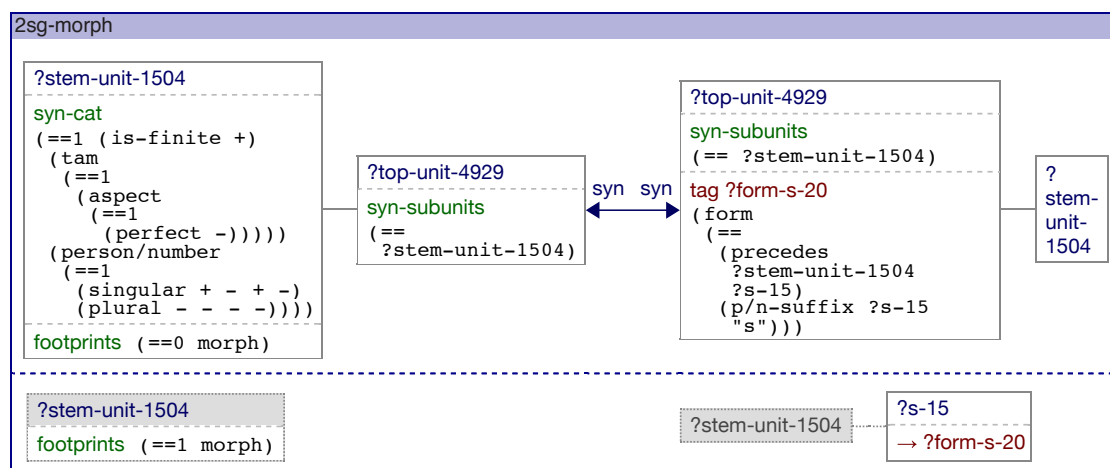


Figure 4.6 – The 2sg-morph construction translates the syntactic categorization of the left pole (2nd person singular) into an agreement suffix that follows the stem: -s.

the production of the -s are specified on the left pole of the construction: the verb stem should be a finite verb, it cannot be past perfect and the person/number feature should be set to 2sg. Finiteness is conveyed by the boolean `is-finite` feature. The `tam` feature indicates that the perfect aspect is excluded because past perfect endings are never split up into TAM and agreement endings but saved as monolithic endings in the grammar. The `person/number` feature matrix is read as follows: the first value after the feature name (e.g. singular) defines whether this feature is expressed (thus whether the morpheme expresses singular for instance) and the remaining three features express the value of the singular feature when it is expressed (1st, 2nd or 3rd person). In parsing, the *2sg-morph-cxn* fires when the FCG parser detects an -s suffix that is preceded by a stem unit (see the `precedes` feature in the right pole).

4.2.4 Phonological constructions

The final construction type that is present in the Spanish FCG grammar deals with the lowest level of linguistic description: phonology. Phonological constructions (or *phon-constructions*) make changes inside the form strings of a construction, on the level of their phonological segmentation. They are also solely operational on the syntactic pole of the transient linguistic structure, where they map certain functions into forms, similar to morph-constructions. The changes that they make inside a verb form depend on the syntactic categorization of the verb and its endings, and is often related to the form's affiliation to a particular verb type (expressed through a `footprints` feature in Figure 4.7). Examples of phonological changes are modifications in the vocalic nucleus of a verb stem (e.g. *e* → *ie* as in *tiene*, 'he/she has'), alternations in the post-nucleus (e.g. *g* → *j* as in *cojeron* (< *coger*), they took) or changes in the suffix vowels (e.g. *i* → *y* as

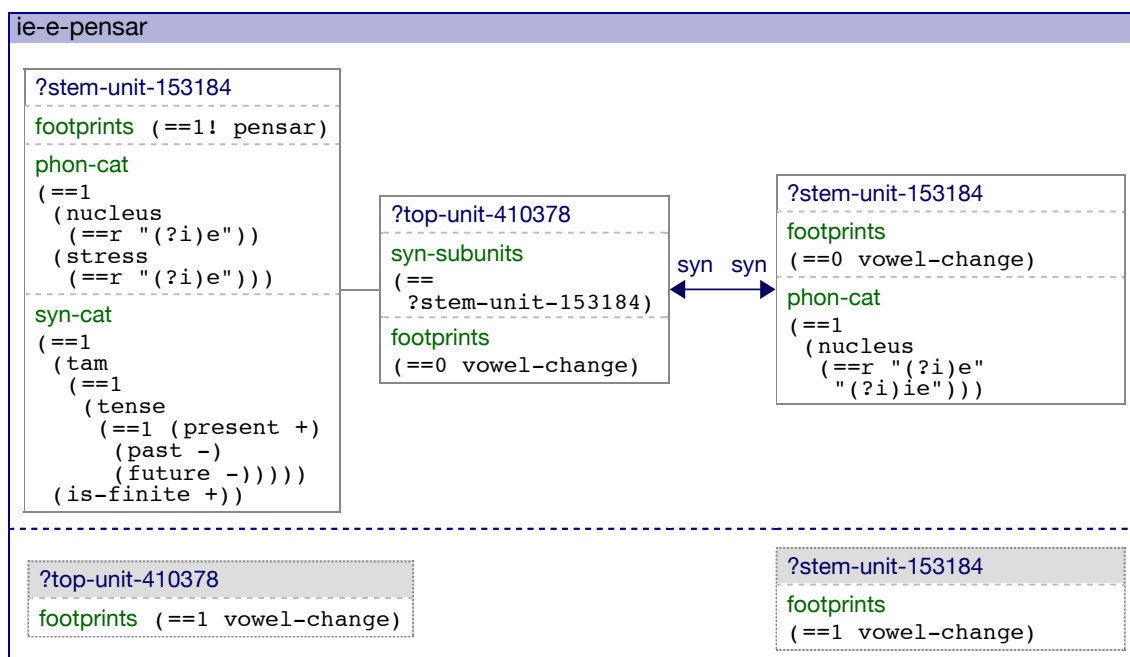


Figure 4.7 – This phon-construction expresses the diphthongization process that occurs in verbs of the pensar-conjugation. When a stem has a stressed "e" as its nucleus, it is replaced with "ie" in production. The \rightarrow operator can override a string with another one.

in *cayó*, 'he fell' or *aba* \rightarrow *ába* as in *hablábamos*). Additionally, phon-constructions also decide on the stress pattern of a conjugated verb form and thus add a **stress** feature to the resulting linguistic structure.

Currently, these constructions work exclusively in production, where they translate from the basic form into a modified form. The left pole of the construction functions here as the condition and the right pole merges new information in when this condition is satisfied. The regular expressions (**==r**) operator takes one element in matching and two elements in merging (see Section 4.3.2). Because this construction expects that there is already syntactic information available about the stem unit, in parsing this is not the case as no other constructions can have applied before the stem change has taken place (the lexical construction matches on the base stem "pens"). The derendering algorithm currently takes care of replacing "pienso" with "penso" (see Section 4.3.2).

After lexical, morphological and phrasal constructions could do their work, phonological constructions **modify** information present in the **phon-cat feature** of the unit they are targeting, given a positive matching result of the syntactic context in which the changes are expected to occur.

Box 4.2 – The first use of phonological constructions in an FCG grammar.

4.3 A grammar engine for Spanish

The basic workings of the FCG interpreter, or the language agent's grammar engine, have been explained in detail in Chapter 3. Section 4.3.1 shows how the grammar engine can conjugate Spanish verb forms by making use of the language agent's construction inventory. The extensions that had to be made to the grammar engine to allow for robust rendering of produced verb forms and verb form segmentation in parsing, along with the introduction of a special operator for regular expressions in the form feature, are described below in Section 4.3.2.

4.3.1 Conjugating verbs with the FCG interpreter

This section considers the actual production and parsing processes of Spanish verb morphology in Fluid Construction Grammar (FCG). Examples for regular, semi-regular as well as irregular verbs are presented. Special cases such as unmarked morphemes and future verb forms are discussed separately at the end of this section. The use of construction sets is demonstrated in this section, with a specific order between them in parsing and production. Inside one construction set, no further ordering is foreseen and all constructions within one set can potentially apply at the same time. Yet, given that the FCG search engine follows a best first search algorithm, the first applied construction will be explored first.

Regular verb conjugation

All verbs whose conjugation is not characterized by the presence of morphophonemic stem changes or holistic verb forms in some parts of their paradigm count as regular verbs. A holistic verb form is a form that cannot be further segmented into a verb stem and a verbal ending such as *son* 'they are' or *voy* 'I go'. Regular verbs follow one of the three main verbal paradigms in Spanish, depending on whether their infinitive ends on *-ar* (1), *-er* (2) or *-ir* (3). An example of such a regular verb form is the second person present form of the verb *hablar*, to speak: *hablas* 'you speak'. The production process that leads to this utterance starts off with the application of the lexical construction of the verb and runs through phrasal and morphological constructions to end with a phonological construction that assigns word stress to the stem, the default stress pattern in this case (Figure 4.8).

Some Spanish verb endings are syncretic, which means that they result in an ambiguous verb form that leads to two possible parsing solutions. An example is the ending *-amos*, which can refer to either:

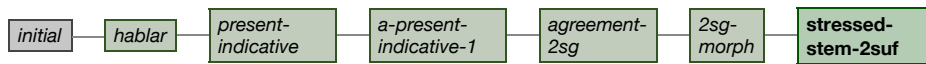


Figure 4.8 – Seven constructions apply in the production process that results in the utterance *hablas* (you speak). In their order of appearance, the following construction sets occur: lexical (*hablar*), phrasal (*present*, *agreement-2sg*), morphological (*2sg-morph*, *a-present-indicative-1*) and phonological (*stressed-stem-2suf*).

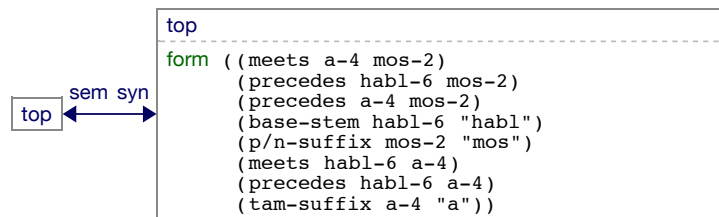


Figure 4.9 – The initial structure of the parsing process of *hablamos*, segmented as *habl-a-mos* contains three strings (one per segment) and five ordering constraints, both *meets* and *precedes* conditions. *Meets* constraints specify direct adjacency relations (e.g. *a* should be directly placed before *mos*). Looser *form* constraints are achieved by so-called *precedes* features (cf. second but last feature in *form* list), where there can be other items between the two argument strings (e.g. the stem *habl* precedes *mos* but does not *meets* it).

1. 1st person plural present indicative, where *-a* refers to the tense and the mood information and *-mos* expresses person and number;
2. 1st person plural past perfect indicative, where *-amos* cannot be separated as such.

Therefore, a verb form such as *hablamos* can thus be segmented as *habl-a-mos* 'we speak' or *habl-amos* 'we spoke'. The segmentation algorithm (see Section 4.3.2) returns two initial structures, which will both be parsed by the FCG engine. Figure 4.9 illustrates an example segmentation for the *hablamos* verb form where it is segmented into three substrings: a *base-stem*, a *p/n-suffix* and a *tam-suffix*. The difference between *habl-a-mos* and *habl-amos* in terms of initial feature structures leads to an *activation of different morph-constructions* during parsing (Figure 4.10): *amos-past-perfect* vs. *a-present-indicative-1* and *1pl-morph*. It is these *morph-constructions* that further select the phrasal TAM-construction that fits the segmentation: *present* or *past perfect*. At the end of the parsing process, two *phon-constructions* try to put stress on the stem but they fail in doing so: not the stem but the *a* in *-amos* receives the main word stress in this verb form.

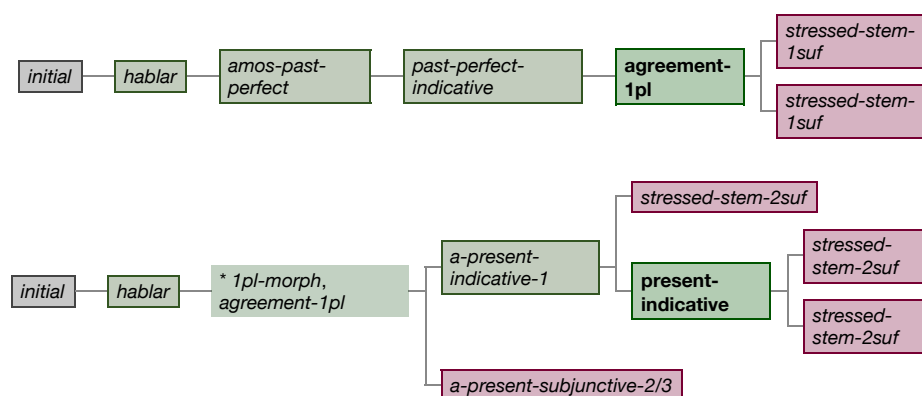


Figure 4.10 – Parsing search process when parsing the ambiguous verb form *hablamos*, 'we talk/ we talked'. The upper subfigure shows the parsing of the segmentation *habl-amos*. The lower one starts from *habl-a-mos* (Figure 4.9).

Table 4.2 – The present indicative conjugation of *pensar*, to think, contains four stem changes: all singular persons and the 3rd person plural are affected by the stress pattern. The diacritic stress is added here for clarity.

piénso
piénsas
piénsa
pensámos
pensáis
piénsan

Semi-regular verb conjugation

Semi-regular verbs are characterized by morphophonemic changes that occur in their stem or suffixes. The full list of changes is included in Section 4.1 above. The current section shows examples of both types of changes. In certain verbs, such as *pensar* 'to think', stem vowels are diphthongized when they receive stress in pronunciation (Table 4.2). In such cases, all singular verb forms and the 3rd person plural form are affected by a stem change due to a particular stress pattern. Because the alternation is lexically arbitrary, not all verbs that have a stressed stem vowel are diphthongized. Minimal pairs such as *contar/montar* 'to count/to mount' differ in their diphthongization patterns: *cuento* for *contar* but *monto* for *montar*. A rule of thumb that is often used to determine the diphthongization pattern of a verb is that when the verb has a nominal counterpart that is diphthongized, the verb follows the same pattern: e.g. *un cuento* 'a story' → *cuento* 'I speak'.

The operationalization of these stem changes is done by a single construction that applies at the end of the processing pipeline during production (Figure 4.11). Since not all verbs diphthongize their stems, all lexical stems are marked with a so-called footprint that indicates the verb group. Some verb groups are characterized by certain stem changes,

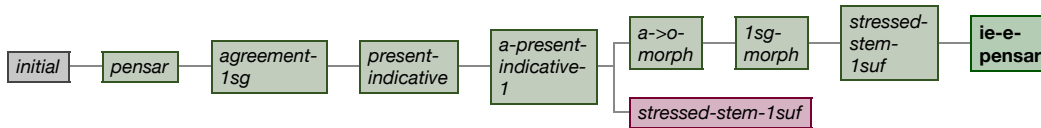


Figure 4.11 – The stem-change construction applies last in the processing pipeline (production). The application of the stressed-stem construction adds a stress feature to the phon-cat of the pensar stem. The default -a- TAM ending is replaced by -o due to the presence of the first person singular.

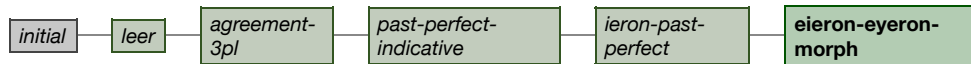


Figure 4.12 – The assimilation between the end vowel of the stem /e- and the past perfect ending *ieron* lead to the suffix change *ieron* into *yeron* (production).

others are not. The **ie-e-pensar-construction** only applies for verbs of the pensar group, whose stem is stressed (Figure 4.7). The verb groups are assigned automatically when a lexical construction is created, following irregularity patterns described by Rello & Basterrechea (Rello & Basterrechea, 2011, p.232).

Morpho-phonological changes can also occur in the suffixes that are attached to the stem. Such changes are mostly due to assimilation processes. They occur when two vowels meet: the stem vowel and the first vowel of the ending: e.g. *cayó* (< *ca-íó*, 'he/she fell'), *leyeron* (< *le-ieron*, 'they read'), etc. Also here, the morpho-phonological change occurs at the end of regular processing (Figure 4.12).

Irregular verb conjugation

Completely irregular verb forms are added to the lexicon so that they can be accessed in their full form. Apart from the infinitive predicate meaning, these irregular lexical constructions also carry semantic and syntactic information about the person, number and tense of the event. Both in parsing and production, the irregular construction is the only construction that applies (Figure 4.13). No morphological, phonological or phrasal constructions are needed since all formal information is contained inside this irregular construction.

Not surprisingly, when processing is taken into account there is one constraint that needs to be considered. In verb paradigms that contain irregular and regular forms (e.g. *valer* 'to be worth': *valgo* 'I am worth' vs. *valemos* 'we are worth') it is important to specify the precedence of the irregular construction set over the default lexical construction set in production so that utterances such as **valo* 'I am worth' are not erroneously produced. Irregular constructions should thus always be applied first, before lexical constructions are tested for matching. If no irregular construction can be found to express a particular

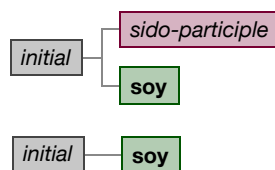


Figure 4.13 – Only a single construction applies during parsing (top) and production (bottom) of the irregular verb form *soy* 'I am'.

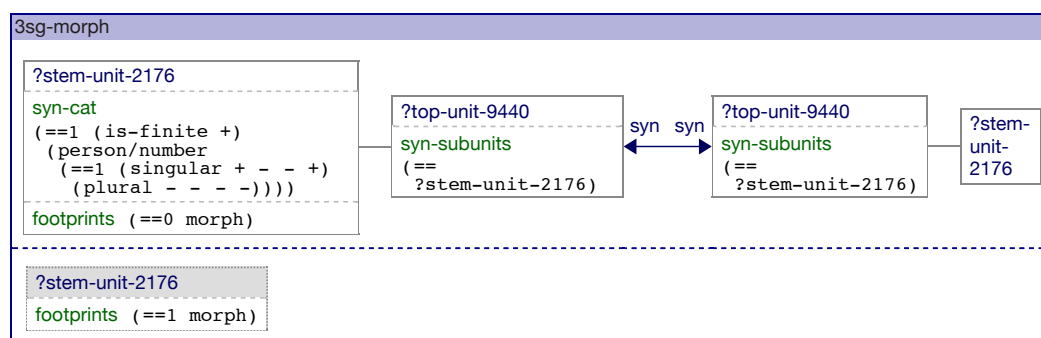


Figure 4.14 – The unmarked morph-construction for the 3sg person/number morpheme can only be used when no other p/n-morph-construction has applied yet (footprints ==0). In parsing, it matches on a stem-unit without further specifications; in production a person/number feature is required, together with a confirmation of the stem's finiteness.

meaning, the default lexical construction for the lemma will trigger.

Special cases: Unmarked morphs and future tenses

There are two special cases that do not obey the default processing behaviour for Spanish verb forms: unmarked forms and the future tense.

- **Unmarked morphemes** occur in the first and third person singular of the conjugational paradigm. They can be categorized as agreement (person/number) morphemes and they are preceded by a TAM-morpheme, which situates the verb form in terms of tense, aspect and mood. The main function of such an unmarked morpheme is to assign a **person/number** feature to a verb form (Figure 4.14). An example is *habla* ('he/she speaks'), which lacks an overt morpheme for the agreement feature but is instead marked with a zero morpheme: *habl-a-∅*. For a detailed explanation on the treatment of unmarked forms in FCG, the reader is referred to the recent volume *Design Patterns in FCG*, chapter 8 (Beuls, 2011, pp. 258-263).
- **The future tense** is also a special case. Because a future verb stem is different from the one that is used for present and past verb forms (e.g. *habl-a* 'he/she

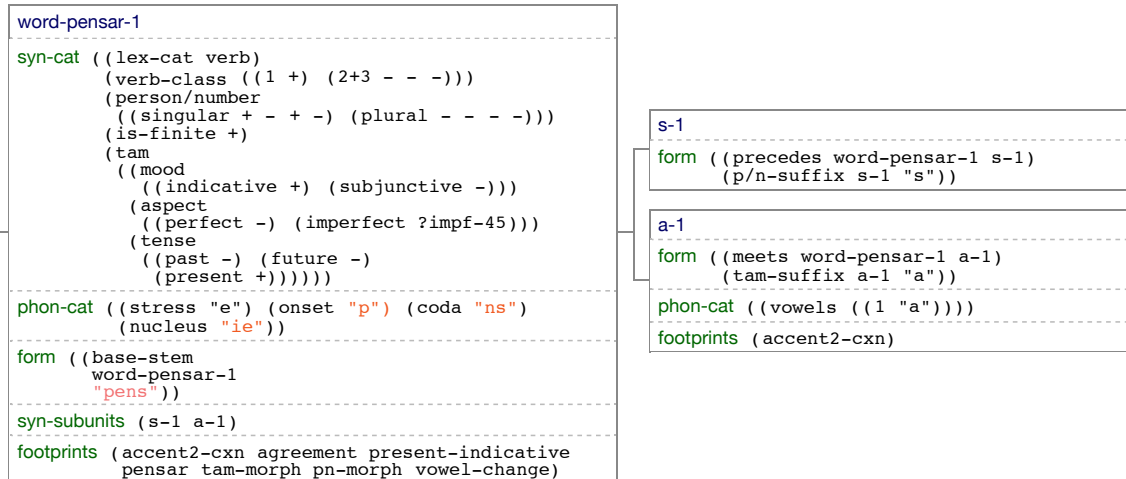


Figure 4.15 – If this structure would be rendered as it is here, the utterance would be *pensas* instead of *piensas* 'you think'.

speaks' vs. *hablar-á* 'he/she will speak'), an additional future stem construction is used to insert the future extension: *-ar* (1st verb class), *-er* (2nd verb class) or *-ir* (3rd verb class).

4.3.2 FCG extensions

Three major extensions were made to the standard FCG implementation to allow for stem modifications and robust parsing: (i) a robust rendering method was created, (ii) a segmentation step was added before the initial feature structure is generated and (iii) a new special operator `==r` that can handle regular expressions in the form feature was introduced. This section discusses each of them in turn.

Rendering modified stems

On the one hand, conjugating an irregular or a semi-regular verb often involves modifying a verb's stem (its main vowel, last or first letter: e.g. *yerro* from *errar* 'to fail'). In technical terms, the phonological constructions that generate these changes only modify the **phon-cat** feature in the syntactic pole of the linguistic transient structure, leaving the **form** feature intact (Figure 4.15). When a transient structure is rendered (after the production goal tests have succeeded (Bleys et al., 2011)), however, these changes are ignored because the standard render algorithm does not consider the **phon-cat** feature. To solve this problem, I implemented a special render method **render-robust**, which transforms the attributes of the **phon-cat** feature into a new form feature. The correct order of these attributes (e.g. **onset**, **nucleus**, etc.) is specified in in the grammar's

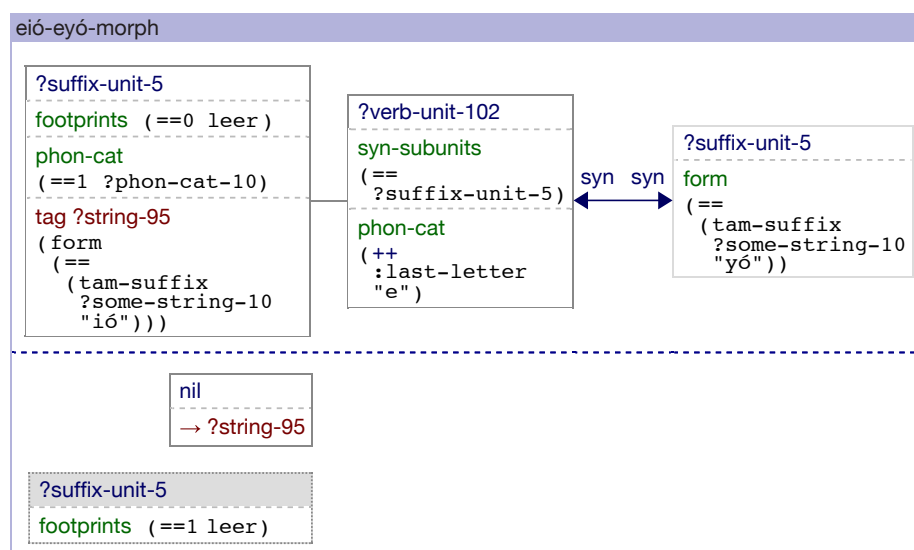


Figure 4.16 – This construction is responsible for the *e-ió* → *e-yó* assimilation process. It is used in the production of verb forms such as *leyó* 'he/she read' and other verbs of the *leer* family.

configuration. The `stress` attribute is ignored in the render process. After the form feature has been replaced, the new linguistic structure is passed to the standard render function.

On the other hand, when a verbal ending is modified due to assimilation or other morpho-phonological processes, this does not affect the renderer. The phonological constructions are created in such a way that the form feature is immediately replaced with a new string. This is done by means of a so-called "nil unit". The original suffix string is hereby moved into this unit in the first merge phase, whereafter a new suffix string is merged into the empty form feature during second merge (Figure 4.16).

The robust-render method scans the resulting linguistic feature structure and translates possible changes that have occurred in the phonetic category of a unit into a string feature that can be rendered as part of a verb form.

Box 4.3 – The robust-render method

Segmentation

The standard FCG parsing module does not contain a stemming or segmentation algorithm. Complex strings need to be pre-segmented by the user before the FCG engine can handle them. A verb form such as *hablas* 'you speak' is thus passed as a list ("habl" "a" "s") to the FCG de-rendering process before parsing can be launched. This list creation

cannot be done manually if you want to allow large-scale grammars for verb conjugation. This section describes the segmentation algorithm that was added to FCG to allow for a direct parsing of verb forms such as *hablas*.

A segmentation algorithm can generally encounter two types of input: grammatical or ungrammatical. A verb form is grammatical when it can be segmented according to the grammar rules that the algorithm has at its disposal. When this segmentation fails, the verb form is either ungrammatical or the grammar needs to be expanded with a new rule. Such a failure manifests itself usually in three ways:

1. Grammatical verb endings are detected but the stem that precedes these is unknown to the system.
2. The segmented verb stem is grammatical but the remainder of the verb form is untraceable.
3. No grammatical segmentation whatsoever is possible. The algorithm does not detect a regular stem or endings.

Let us now run through an example. The segmentation process of the grammatical verb form *abandonamos* 'we abandon(ed)' proceeds as follows:

1. The form is first segmented into a list of possible stems (*abandon*) and a list of possible endings (per stem) (*amos*). An ending is defined here as the letters that succeed a particular stem, without any further interpretation.
2. Based on these two lists combinations of stems and suffixes are created to verify whether they can form realistic verb stems. In the case of this example there is only one combination possible and it is equal to the original verb form: (`"abandonamos" ("abandon" "amos")`).
3. A number of conditions are checked and further segmentations are made:
 - The input form is an infinitive → return it as such
 - The input form is an irregular form → return it as such
 - The preliminary segmentation leads to the original verb form → return segmentation with preliminary stem and segment suffix further: (`("abandon" "amos" NIL) ("abandon" "a" "mos")`)
 - When there was no valid combination possible the stem and suffix lists are explored and all meaningful segmentations are returned. There is an additional

heuristic that decides which segmentation to select when there is more than one. This heuristic parses and reproduces the segmentations and checks the results.

Regular expressions

Regular expressions allow you to formulate text search strings. First developed by Kleene (Kleene, 1956) in 1956, a regular expression "is a formula in a special language that specifies simple classes of strings" (Jurafsky & Martin, 2009, p. 52). Regular expression search typically involves a pattern that is searched for and a corpus of text to search through. Regular expressions are case sensitive, which implies that `/l/` is different from `/L/`. Matching the string *Luc* with the pattern `/luc/` will thus fail. Allowing the pattern to match a string regardless of its initial capitals requires the use of the square brackets: `/[lL]uc/`.

The FCG engine does not perform a string search but strings are compared during matching and merging of the `form` feature. This comparison is by default an exact match. The feature attribute `(string ?luc-unit "luc")` will match `(string ?noun-unit "luc")` but not `(string ?noun-unit "Luc")` or `(string ?noun-unit "Luciano")`. However, some constructions that are involved in verb conjugation may be useful for more strings than the ones exactly specified in the `form` or `phon-cat` features. For instance, both the verbs *sacar* 'to receive, to learn' and *acercar* 'to approach' have a *-qu-* stem ending when the stem precedes an *é* or *e*. Because their codas are different (*-c-* vs. *-rc-*) the standard FCG processing requires two separate constructions: one that changes *-c-* into *-qu-* and another one for the change from *-rc-* to *-rqu-*.

To avoid a duplication of constructions I introduced a new FCG special operator that can match and merge regular expressions: the `==r` operator. The operator is linked to a match function and a merge function that rely on two functions of the CL-PPCRE library (Portable Perl-compatible regular expressions for Common Lisp): `all-matches` and `regex-replace`. The `all-matches` function returns a list with match positions that indicate the start and the end locations of a pattern in a particular string. For instance, the list `(0 1 3 4)` indicates that the pattern was matched from position 0 to 1 (first letter) and from position 3 to 4 (fourth letter): the result of searching the pattern `/e|i|í|é/` in the string *iste*. The `regex-replace` function searches for a pattern in a string and replaces the matched substrings with another string. To return to the previous example, the pattern `/*c/` could be replaced with `/*qu/`, with the asterisk representing any sequence of letters (or an empty string) preceding *c* or *qu*.

The new `==r` operator is used at the beginning of a list that replaces the string in a `form` or `phon-cat` feature. For instance, `(coda (==r "(?i)c"))` instead of the usual `(coda "rc")`.

4.4 Related approaches in computational morphology

There have been many previous attempts at formalizing verb conjugation, be it rule-based (e.g. by means of finite state transducers (Beesley & Karttunen, 2003; Koskeniemi, 1983b)), through a supervised learning task (e.g. memory-based learning (Van den Bosch & Daelemans, 1999; Keuleers & Daelemans, 2007)) or in a fully unsupervised fashion (through probabilistic methods (Creutz & Lagus, 2002; Snyder & Barzilay, 2008; Spiegler, Golenia, & Flach, 2009)). Although Construction Grammar is an ideal framework to think about inflectional morphology, most research in Construction Morphology focuses on word formation and derivational morphology (Booij, 2010, p.23). However, inflectional morphology has the advantage that it covers very large morphological paradigms that often show a discrepancy between regularity and irregularity (Corbett, 2003), which are interesting for formalization attempts. This section compares my proposal of Spanish verbs in FCG with finite-state morphology approaches to morphophonological changes and points to previous proposals for inflectional morphology within the computational Construction Grammar community.

4.4.1 Finite-state morphology

The origins of finite-state morphology can be found in the ordered phonological rewrite rules of traditional phonological grammars, formalized by Chomsky and Halle (1968). Such rewrite rules were used to translate abstract phonological representations into surface forms through a series of intermediate representations. They have the general form $\alpha \rightarrow \beta / \gamma _ \delta$, where α , β , γ and δ can be arbitrarily complex strings or feature matrices (Karttunen & Beesley, 2005). Yet, these sequential rules were not as powerful than was first thought. Johnson (1972) showed that they can be modeled by finite-state transducers, which by definition represent regular relations. Moreover, rewrite rules were exclusively used for generation and it was not evident how they could lead to an efficient morphological analysis (going from surface to lexical forms). A deterministic generation would often lead to a nondeterministic analysis with multiple possible lexical forms for a single surface form. This asymmetry "is an inherent property of the generative approach to phonological description" (Karttunen & Beesley, 2005, p. 73).

Two-level rules were introduced to avoid the asymmetry problem in one-directional ordered rewrite rules. Two-level morphology, as introduced by Koskeniemi (1983a), formalize the lexicon itself as a finite-state transducer and compose it using rules. Rules are statements that contain surface constraints directly. According to Karttunen and Beesley (2005, p. 74), two-level morphology is based on the following three ideas:

1. Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially

like rewrite rules.

2. The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.
3. Lexical lookup and morphological analysis are performed in tandem.

Of course, applying the rules in parallel does not in itself solve the nondeterministic analysis problem. Yet, the key in two-level morphology lies in the avoidance of an intermediate level of analysis. In Koskenniemi's first system, "the lexicon was represented as a forest of tries (= letter trees), tied together by continuation-class links from leaves on one tree to roots of another tree or trees" (Karttunen & Beesley, 2005, p. 76). This approach was the first practical general model for the analysis of morphologically complex languages. Language-specific components, such as the rules and the lexicon, were combined with a universal runtime engine applicable to all languages.

My FCG implementation of verbal inflectional morphology works solely with symbolic rules and the lexicon contains only regular surface forms but is not informed about variants of theses. The constraints for changing stems or suffixes are formulated on the left-pole of a morpho-phonological construction, while the right pole of this construction instantiates them. The constraints refer to phonological features (stress, nucleus) and syntactic features (tense, aspect, mood, person, number). It is thus different from two-level morphology in that it does have an intermediate level in the application pipeline, where lexical forms have not yet been "finalized" as surface forms.

4.4.2 Construction Grammar morphology

Other computational Construction Grammar frameworks such as Embodied Construction Grammar (ECG) and Sign-based Construction Grammar (SBCG) have also previously looked at inflectional morphology, and suggested theories to handle parsing of inflected verbs. Nathan Schneider has demonstrated how Hebrew verb morphosemantics can be captured in ECG by means of verb schemas and a range of constructions (stem constructions, inflectional constructions, etc.) (Schneider, 2010). The ECG approach allows morphological complexity to be expressed in several ways such as stored representations (e.g. *goes*), morphological constituency (e.g. *jumps*: Jump and Plural-Suffix), inheritance from abstract representations (e.g. *jumps*: Jump schema and third person Singular-Present-Tense-Verb construction) or morphological maps which take semantic and formal structures of a given word and map them to morphologically related words (e.g. *give* → *gave*) (Bergen, 2003). In SBCG, by contrast, morphological representations are established through theoretical principles. These principles are typically integrated in morphological functions, which define the relation between the form of a lexeme (e.g.

laugh) and the form of a verb form (e.g. *laughed*) (Sag, 2012). They are then used by inflectional constructions (e.g. Preterite Construction) that contain a predefined verb form feature with a fixed set of possible values.

While ECG and SBCG are restricted to the use of parsing, FCG exploits its bidirectional architecture to self-diagnose potential problems. Full or partial parsing solutions can be compared to the utterances that the production of these solutions would yield¹. I also opted for multiple representations of morphological complexity, ranging from irregular lexical constructions (e.g. *was*), morpho-phonological constructions that alter the verb stems up to highly regular morphological constructions that provide verb endings. My FCG Spanish grammar contains specific constructions for each verb type and more general inflectional (tense, aspect, mood), argument structure and morphological constructions. As Goldberg emphasizes in her 2006 monograph: "*it is constructions all the way down*"(Goldberg, 2006, p.20) (original emphasis): from abstract argument-structure constructions to individual morphemes.

4.5 Conclusion

This chapter has demonstrated how to implement an FCG grammar for Spanish verb conjugation that can handle the various complex layers it entails. The construction inventory that supports this language system therefore contains bi-directional constructions to ensure a correct inflection of Spanish regular, irregular and semi-regular verbs in all 19 tense, aspect and mood combinations. Irregular verbs are memorized as holistic chunks that map a full verb form meaning onto a form string that cannot be analyzed further. Semi-regular verbs modify their regular stems through a series of morpho-phonological constructions (one or more).

The implementation of the Spanish verb grammar required the following four ingredients:

1. Four types of constructions: lexical, phrasal, morphological and *phonological*. The latter is a new type that was introduced for the first time in this chapter. It includes the use of a new special operator `==r` that allows regular expressions within features that contain strings.
2. A separation into ordered *construction sets* with different orders in production and parsing. This order is essential to correctly produce irregular forms or assign word stress.

¹ I prefer the use of the term *production* instead of *generation* since the grammar processor does not generate all possible solutions but only these that fit within constraints imposed by knowledge of the situation, preferences of the language user, frequent language patterns, etc.

CHAPTER 4. SPANISH VERBS IN FCG

3. A *robust rendering method* that translates a final linguistic structure (including morphophonemic changes in the phon-cat) into an utterance at the end of production.
4. A *segmentation algorithm* that segments a verb form into its functional morphemes: stem + one or two endings.

Yet, the constructions that were shown in this chapter had all been written by hand so that constructing a large construction inventory is not a feasible task. Because our language agent needs to have a realistically sized-grammar, a special algorithm has been designed that creates all constructions needed to conjugate one specific verb lemma when you give it an infinitive. This automatic conjugator is explained in Chapter 8 where it is compared with incremental learning strategies that are used by a student agent in situated learning settings. The following two chapters focus on how ungrammatical verb forms can be parsed with the flexibility strategies that a language agent is equipped with.

Chapter 5

Flexibility strategies

Although the grammar engine and the construction inventory of a language agent might be optimised for fast and efficient processing, language input is often far from perfect. Small linguistic deviations in a conversation can be due to innovative language use on behalf of the speaker, marked pronunciations that reveal the speaker's language community or careless language that is manifested in broken syntactic patterns or ambiguous semantic expressions. Similar to a competent language user, the language agent needs thus to be equipped with the capacity to parse and produce in a more flexible and robust way so that unexpected processing failures can be avoided.

I therefore rely on the well-established idea in computer science of *computational reflection*, a concept that was first introduced in the PhD thesis of Brian Smith (B. C. Smith, 1982). Analogous to how you can think about your own thinking, a computer program can also reflect about its actions so that errors can be caught in time or planning and learning can take place. For computational language processing, a similar architecture can be reused that employs regular language processing on the data object level and meta-linguistic operations in a second processing level. The current chapter explains this *meta-level architecture* and pays special attention to the representation of *flexibility strategies* that are active in this additional processing level. These strategies are vital to build a reliable model of a competent language agent.

Flexibility strategies do not change the construction inventory or the grammar engine in a permanent way but can temporarily relax certain grammar constraints or alter pre-specified stopping criteria in the search tree. They are therefore ideal for identifying learner errors and performing conversational repairs. This chapter is structured as follows: First, Section 5.1 explains the concept of computational reflection in further detail, connecting the use of the meta-level architecture in this dissertation to its original application. Section 5.2 describes the formulation of flexibility strategies in terms of

diagnostics and repairs that catch problems during linguistic processing. Finally, Section 5.3 demonstrates the use of FCG for performing conversational repairs.

Flexibility strategies do not change the construction inventory or the grammar engine in a permanent way but can temporarily relax certain grammar constraints or alter pre-specified stopping criteria in the search tree.

Box 5.1 – The power of flexibility strategies.

5.1 Computational reflection

The concept of computational reflection was first introduced in the PhD thesis of Brian Cantwell Smith in 1982 in the context of procedural programming languages. He argued for "a theory of *procedural reflection* that enables any programming language to be extended in such a way as to support programs able to access and manipulate structural descriptions of their own operations and structures" (B. C. Smith, 1982, p. 3). The term *reflection* finds its origins in the field of logic where it was used to "denote a way of extending theories" (Maes & Nardi, 1988, p. vii). It has later been applied not only to logic-based reasoning systems but also to cognitive architectures such as SOAR (Laird, Newell, & Rosenbloom, 1987)) and as a general powerful programming language construct in Object Oriented Programming (OOP). Also operating systems are today standardly built with a reflective component, so that complete system crashes can be avoided as much as possible.

In another MIT dissertation, Batali (1983) talks about *computational introspection*, with which he means the same as Smith's reflection, and lists some introspective activities that are crucial for intelligence and understanding representations. These activities involve "some sort of access to, or modification of, representations of either the processor itself or representations of its actions, skills or abilities" (pp.14).

1. Learning: an agent can modify itself as a response to certain activities in such a way that these changes affect later similar situations. Learning usually require changes that improve the behaviour of an agent in some dimension.
2. Planning: the elements in a plan represent the sequence of activities that an agent will perform and are thus introspective in their representation of the agent.
3. Advice taking: an agent must be able to improve its actions if someone tells it explicitly what should be done.

4. Assumptions: an agent must be able to deal with an assumption if it turns out to be incorrect.
5. Serendipity: agents must be able to realize that their goals are met even if it happens by accident.
6. Defaults: contrary to assumptions, defaults are asserted unless they are specifically overruled.
7. Debugging: the concept of failure requires an understanding of goals. But also the fixing of failures is an introspective skill, as you cannot fix something unless you know how it works.
8. Efficiency: only when a program would have access to representations of the computational capabilities of the machine it is running on, problems can be solved efficiently.

In a *reflective architecture*, a computational system "is viewed as incorporating an object part and a reflective part" (Maes, 1988, p. 23). Object computation is thought to "solve problems and return information about an external domain", while "the task of the reflective level is to solve problems and return information about the object computation". Reflective architectures are today mirrored in ideas of of autonomic computing. To be autonomic, a computing system needs to "know itself" – and comprise components that also possess a system identity. A technical report by IBM sees autonomic systems as a Grand Challenge for the entire IT industry:

We'll need to make progress along two tracks: making individual system components autonomic, and achieving autonomic behavior at the level of global enterprise IT systems. (IBM, 2001, p. 30)

Meta-level architectures have thus established a crucial concept in computer science ever since their introduction in the 1980ies. They form the real hallmark of robust computational systems. Their functionality can directly be extended to programs for speech and language processing, relying on computational reflection to deal with uncertainty and problems that might cause crashes of the routine processing pipeline. The next section discusses their use for robust language processing through the introduction of flexibility strategies, with a single strategy for every processing problem that can be encountered.

5.2 Meta-level flexibility strategies

A reflective, or meta-level, architecture for language processing is responsible for the robustness of the grammar engine and the construction inventory that is plugged into the engine. But there are many ways in which robustness can be guaranteed during language processing. One language agent could relax the matching or merging constraints to let a construction apply to a certain transient structure, while another language agent could instead deliberately modify the features of a construction that hinder its application. To capture such varying solutions to one particular linguistic processing problem that a competent language agent encounters, I introduce the notion of a *flexibility strategy* here. A flexibility strategy intends to guarantee flexible language processing and can be seen as a plan to tackle a problem, which includes diagnostics to detect it and repairs that try to solve it.

One thing that flexibility strategies have in common is the notion of a *problem*. This notion is wide-spread in the fields of artificial intelligence, mathematics, psychology and medicine. In AI, problem-solving algorithms are methods (generic or ad-hoc) to find solutions to problems that manifest themselves inside a computerised process. Problems are typically categorised as ill-defined or well-defined. Ill-defined problems do not have clear goals or solution paths, whereas well-defined problems have specific goals and well-defined solution paths. The use of the term problem in this dissertation is restricted to a particular data structure that is located in the linguistic meta-level and that can be identified by one or more diagnostics and solved by a single or multiple repairs. Table 5.1 summarizes the fields that a meta-level problem typically has.

A flexibility strategy tackles *one processing problem* that a language agent might encounter by making use of a number of *diagnostics* and *repairs* that are specialized for that problem.

Box 5.2 – Problem-solving strategies

A single flexibility strategy incorporates all the diagnostics and repairs that treat a single problem (see Figure 5.1). The use of the word "strategy" refers here of the deliberate

Table 5.1 – A processing problem that is identified by meta-level operators has obligatory the following fields.

identifier	the problem's id
priority score	used to rank problems in the meta-level
triggered-by	diagnostics that triggered the problem
solved-by	a list of repairs that have successfully solved the problem

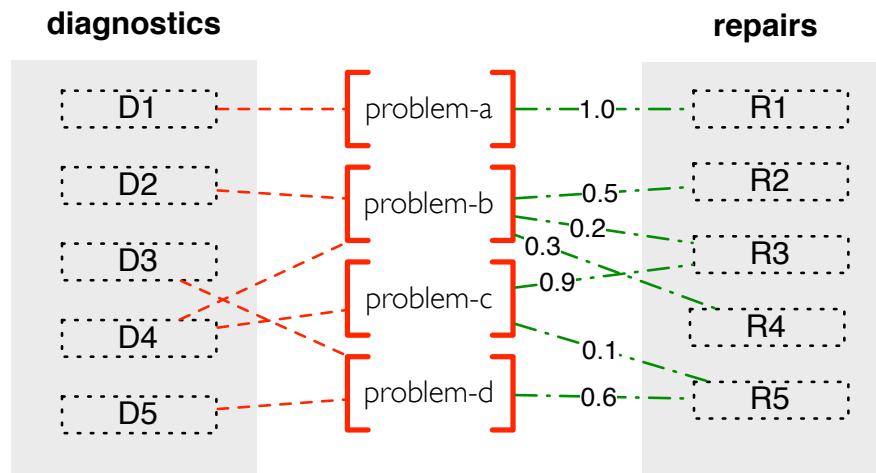


Figure 5.1 – A flexibility strategy tackles one processing problem that a language agent might encounter by making use of a number of diagnostics and repairs that are specialized for that problem. Diagnostics are represented here as red striped lines (D1-D5). Once a particular problem has been diagnosed, repairs do the real work by testing possible solutions of the problem (R1-R5). As this figure shows, one diagnostic can trigger multiple repairs and one repair can be triggered by multiple diagnostics.

choice that is made between a number of repairs that can solve a diagnosed problem but also incorporates the order of diagnostics that were triggered to diagnose it in the first place. As the outcome of a strategy is by default unknown, a strategy does not always succeed and often needs to be adapted to every new use case. A *diagnostic* is a tool that monitors regular language processing and detects irregularities or unforeseen circumstances that might occur. After a thorough examination it can identify and signal a potential problem. A *repair* is triggered by a diagnosed problem and reacts by trying out a solution that is fit for the current situation that the problem was diagnosed in. A single problem-solving strategy can thus consist of a range of diagnostics that can each trigger one or multiple repairs (Figure 5.1). I will refer to *DR-pairs* to verbalize the relation between a particular diagnostic and a repair that solve a problem: e.g. D2-R3 solve problem-b in Figure 5.1.

The edges between problems and repairs are typically scored, as can be seen from Figure 5.1. These scores express the probability that a language agent with this learning strategy will use a certain repair. When there are multiple repairs for one problem, for example as in problem b in the figure, there is a 50% change that repair 2 will be used, 30% for repair 4 and 20% for repair 3. When the probability scores for repairing a particular problem do not sum up to 1.0, this means that there is a certain probability that the problem does not get repaired after all (e.g. problem d has a 60% change of being repaired).

In a competent language agent, flexibility strategies are executed in an additional processing layer, which is thought to operate on top of routine language processing. Routine

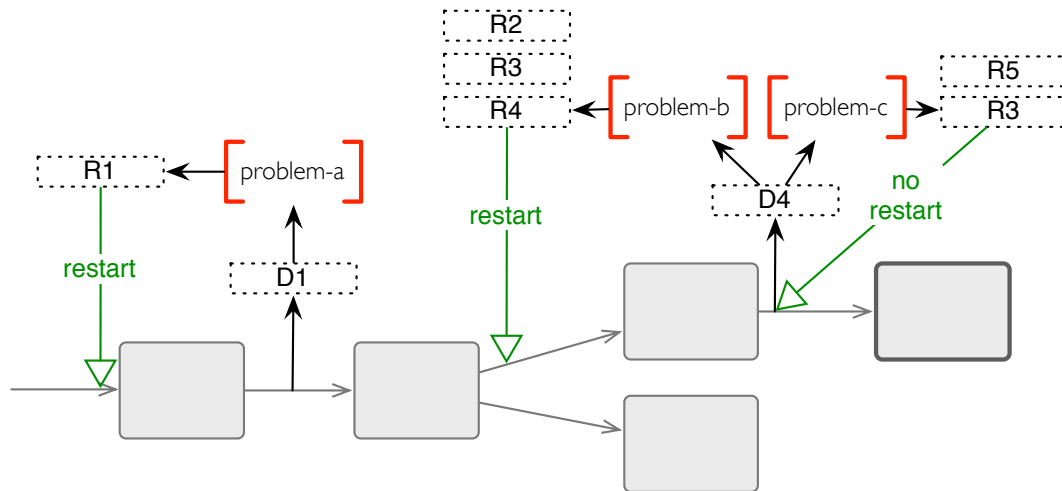


Figure 5.2 – Diagnostics and repairs operate on top of the routine constructional processing layer. Diagnostics can signal problems if unexpected language processing is detected. A problem can be handled by a range of repair strategies, which can potentially trigger a restart when they repaired the problem.

language processing is the object level in which constructions are applied onto an existing linguistic structure with the goal of parsing a given utterance or producing a conceptualized meaning. In this meta-level architecture diagnostics are like daemons, continually monitoring parsing and production processes so that any minor inconsistency can be diagnosed as early as possible (or can be delayed if more information is required). Specialized repairs are triggered when a diagnostic has identified a problem. They try to solve the problem so that regular processing can be continued without any (visible) interruptions. Figure 5.2 visualizes the meta-level architecture. When a problem is identified and multiple repairs could apply, one is selected and evaluated under the given circumstances. When it fails to repair the problem, the next repair in line is given a chance.

5.2.1 Diagnostics

Diagnostics are tools to perform diagnosis, an action that usually defined as "the identification of the nature of an illness or other problem by examination of the symptoms" (British Dictionary). The origin of the term diagnostic (as well as repair) as I use it here dates back a few decades to research on meta-level structures (Maes & Nardi, 1988). These architectures allow a direct realization of the "thinking about x", where "x can be a deduction in a logic-based reasoning system, a computation in a programming language or the action of a system upon its environment" (Maes, 1988), but also the processing of a linguistic utterance or a conceptualized meaning. In the field of Artificial Intelligence, diagnosis is concerned with the development of techniques and algorithms that can estimate the accurateness of a system's behavior. In case of a malfunctioning of the system, the diagnosis should determine which part of the system is failing and discern

the type of defect that causes the failure. Diagnosis always relies on experience with the system, which can be used to build up mappings between observations and corresponding diagnoses.

The same line of reasoning applies to language diagnosis. Experience with a particular language system allows you to fine-tune your observational power and the improves the diagnoses you can make to a point where you start hearing the slightest differences in accent between native speakers, occasional inconsistencies in article usage, novel conceptualizations, and so on. Although there has been some (very) experimental work recently by van Trijp (2012), the diagnostics that I will consider here do not evolve over time, nor can new diagnostics be created autonomously by the language agent. All have to be provided by the ITS designer. The only adaptive aspect is their confidence score that determines their preference order.

In this meta-level architecture diagnostics are like daemons, continually monitoring parsing and production processes so that any minor inconsistency can be diagnosed as early as possible.

Box 5.3 – Daemon diagnostics

My implementation is based on an earlier meta-level architecture for language games that could diagnose and repair problems after every process an agent had completed (e.g. conceptualization, production, parsing, interpretation) but not during language processing (Loetzsch, Wellens, De Beule, Bleys, & van Trijp, 2008; Steels & Loetzsch, 2010). A diagnostic is implemented as a data object that can be activated during parsing or production but never in both. You can also specify a learning situation or a list of learning situations as a condition for diagnostic operation to diminish its possible moments of execution. Typical *learning situations* are: after receiving feedback, when selecting a learning topic, after production, etc. When no learning situation is specified a diagnostic will be active throughout a complete tutor-learner interaction (also referred to as a *language game* further, see Section 5.2.3). As soon as the conditions for a diagnostic are met, a special method is called that carries out the actual diagnosis. This method operates on the data structure that it is manipulating at the moment of the call, which can be a node in an FCG search tree, a process in a language game or a turn in the turn-taking pattern in a conversation. Notice that these structures increase in granularity: an FCG node is part of a parsing or a production process and a process is one of a sequence of many that add up to one turn. Because it has been my contribution to bring diagnostics into FCG, I will be referring to FCG nodes as the level of operation for a diagnostic.

The diagnosis method is then written according to the processing problem that it should diagnose. It can extract information from the FCG node, which contains the transient

linguistic structure, the constructions that have been processed so far (and their order of application), the construction inventory, information on problems that might have been diagnosed already and whether they have been repaired, the configurations of the grammar engine and some book-keeping about the search tree (number of nodes, branching, failed nodes, etc.). Most processing problems relate to the application or non-application of certain constructions and their impact on the transient linguistic structure. Specific diagnostics can inspect particular features in the transient structure but also the hierarchy between units might be a trigger for diagnosis.

A diagnostic either returns NIL (when it could not diagnose anything) or the problem that was detected, which is a data object in itself. The problem can contain various fields that you predefine, such as the word that was identified as unknown, the construction that was used in a novel way or the part of the conceptualized meaning that could not be expressed. Occasionally, a diagnostic can also trigger another diagnostic when it diagnoses a subproblem or the "symptoms" that the diagnostic is investigating require further investigation. A sequence of diagnostics can be invoked by defining a learning situation at the point in the diagnostic evaluation where another diagnosis is advisable.

5.2.2 Repairs

We often think of repair as fixing a mechanical or electrical device that is broken or out of order. Yet, repair often also includes routine actions to keep the device working, such as when you bring your car to the yearly maintenance service. Also linguistic repairs have this double function of maintaining the language system in shape as well as fixing processing problems that occur. In the meta-level architecture, a repair can be regarded as a specification that tells you how to solve a particular problem that has been diagnosed. Its success is however not always guaranteed and the fix should be tested to verify this.

Repairs in language often have a double function of maintaining the language system in shape as well as fixing processing problems that occur.

Box 5.4 – Maintenance and repair.

A repair is a data object – similar to a diagnostic – that has four main slots: triggering problems (default NIL), learning situations (default NIL), success score (default 1.0) and processing direction. Triggering problems is a list of all problems that can be solved by the repair. Learning situations serves the same purpose as in a diagnostic and can be left unspecified when irrelevant. A repair's success score resembles its performance and

fluctuates between 0 and 1. The higher, the better. Success scores can be used to rank repairs absolutely or as probability scores to distinguish learner tactics.

The processing direction specifies whether the repair is specific to parsing or production. When the conditions specified by these slots are satisfied, a special method is called that specifies the repair's protocol, i.e. the order of operations that need to be carried out to fix the broken processing node or to guarantee flexible language processing. These operations include actions such as adding a new construction to the inventory, modifying an existing construction by changing some of its features or modifying its success score, tuning the grammar engine configurations or even revising the scores of other repairs or diagnostics. In principle, constructions are never deleted by repairs but their scores might be decreased so much that they are not considered any longer during processing.

Repairs can evoke restarts after they have successfully solved a particular problem. A restart is not only useful to guarantee uninterrupted processing but also serves to immediately verify the effect of the recent repair. When we run into the same problem after repairing, this could be captured by a higher level diagnostic that regulates repairs so that we do not enter a eternal repairing loop. Figure 5.2 shows examples of restarts in an FCG search process. Two of the repairs that trigger carry out a restart, which means that instead of continuing from the point in the process where a problem was diagnosed, we backtrack to an earlier point, which is specified by the restart. When multiple repairs are operating on the same diagnostic (either through one or more problems), the one to issue a restart suspends all ongoing repairs.

5.2.3 Flexibility strategies interacting with language processing

Now that the basic principles behind diagnostics and repairs have been clarified, it is time to look at some examples of how they exactly interact with routine language processing. Imagine the following conversation between a foreign exchange student and the father of an English-speaking host family (taken from Beuls, van Trijp, and Wellens (2012)):

- Example 1**
- Father: Could you pass me the salad, please?
(The student hesitates and then reaches for the salt.)
(The father shakes his head.)
 - Father: No, I meant the *salad*. (Points to the salad bowl.)
(The student puts the salt back and hands over the bowl.)
 - Father: Thank you.

This short interaction is an example of a *language game*, a routinized goal-direction linguistic interaction, in which several problems occur. First, the student experiences difficulties in parsing the word *salad*, and interprets it instead as *salt*, a word that also

happens to be a good fit in the current situation. Yet, the host father notices that his utterance did not reach the desired effect and shakes his head to signal communicative failure. Knowing that the student does not yet fully master the English language, he therefore repeats the word *salad* with more emphasis while pointing at the salad bowl. The student now realizes that he in fact encountered a new word and tries to infer its meaning from the context. Alternatively, perhaps the student already had a salad construction but did not yet get accustomed to its English pronunciation as /'saləd/ ¹.

I will concentrate here on the learning that happens inside the parsing process of the father's first utterance. Given the student's response to it, we can infer that he misunderstood the word salad and went for the closest matching word in his inventory, in phonetic form and situated meaning. Of course, given this short fragment we cannot judge the English language knowledge of the student. Instead of asking for a clarification, he made a guess and reached out for the salt, an object on the table whose name closely resembles that of the requested item. So how can we model this repaired parsing process with diagnostics and repairs?

First, a new diagnostic needs to be defined that detects unknown words. I assume here that the student did not yet have the salad construction in his construction inventory. If he had, this diagnostic could still trigger but the repair would be a different one. The diagnostic **detect-unknown-word** is activated in parsing and no learning situation is specified. A diagnosis method is then defined that runs as soon as this diagnostic is triggered. This method verifies if there are unprocessed strings left in the linguistic structure that is contained in the final node of the search tree. If there is one unknown string, the method instantiates an **unknown-word** problem. For illustration purposes, the diagnostic only handles single unknown words instead of multiple unknown strings. In pseudo code, the method looks as follows:

```

1: if Leaf?(fcgNode) then
2:   unprocessedStrings ← ExtractUnprocessedStrings(fcgNode)
3:   if unprocessedStrings then
4:     return new UnknownWord(unprocessedStrings)

```

Problematic search nodes are colored differently than successful ones in the FCG web interface. They also receive an additional status: **problem-found**. This status can in turn be consulted by node tests or goal tests that define whether a search branch should be continued or not. Figure 5.3 shows a screen shot of such a node, where the top unit (the open box to the right) acts as a buffer that contains all unprocessed information.

¹ In fact, salad really comes from salt when we look at its etymology. The English (and the Dutch) word comes from French, where it was borrowed from the North-Italian *salada*, *salata* 'the conserved, the pickled', originally 'the salted', a nominal use of the past participle of *salare*, *insalare* 'to salt' < vulgar Latin **salare* 'to salt, making salt', which is in turn a derivation of the Latin *sāl* (Philippa, Debrabandere, Quak, Schoonheim, & van der Sijs, 2009)

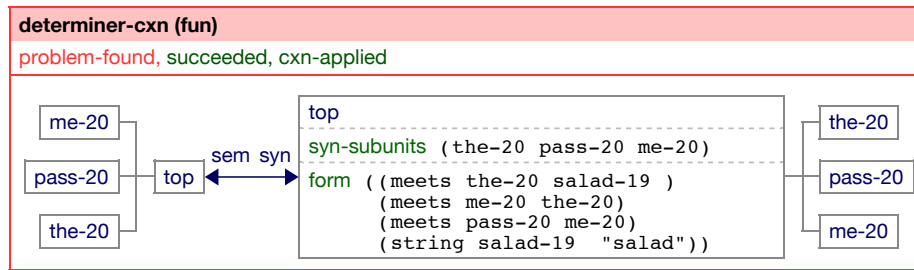


Figure 5.3 – A problem is diagnosed after the string "salad" is left unprocessed at the end of the search tree.

As can be seen, the unprocessed string as signaled by the diagnostic is "salad". Also the word order conditions (cf. `meets` attributes) are still unprocessed at this stage.

Second, once an unknown word has been detected inside the FCG search tree, a repair will trigger and try to solve the problem, which is in this case caused by the word "salad". An example of an FCG repair strategy that tackles this problem is `retry-with-closest-match`. Such a strategy loops through all words in the current grammar and find the word that mostly resembles the unknown word based on its form and the situation that is given. The example repair strategy here only considers similarity in terms of spelling, not in phonetic form. In a more advanced implementation, the latter could of course also be taken into account.

When these initial values are satisfied, a specialized repair method can execute the closest match repair. The pseudo code explains how the original utterance by the host father (expert-utterance) is replaced with a slightly modified version (learner-utterance) by substituting the unknown word with its closest match. The function `find-closest-string` is responsible for searching the existing lexical items and returning the most similar word:

```

1: utterance ← RenderedLinguisticStructure(node)
2: unknownWord ← Word(problem)
3: closestMatch ← ClosestRelatedWord(unknownWord, Lexicon(fcgNode))
4: if closestMatch then return true
5:   revisedUtterance ← ReplaceWord(unknownWord, closestMatch, utterance)
6:   RestartSearchTree(revisedUtterance)
7: else return false

```

When the search tree is restarted, the initial node contains the substituted utterance (see Figure 5.4) and parsing succeeds. Yet, although the processing problem has been repaired, the game still fails since the student did not manage to retrieve the correct object form the context. The student also did not really learn something, that is, in technical terms no new construction was added to the grammar.

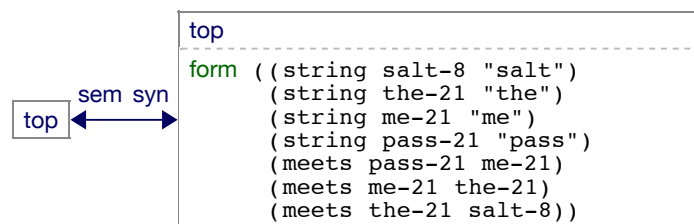


Figure 5.4 – The new initial node after processing has been restarted.

An alternative repair would involve the use of a "generic" construction that takes the unknown word as its form, but which leaves its meaning underspecified. The semantic and syntactic categorization of the word can already (partially) be filled in, as we can infer (through the application of the ditransitive pass construction) that we are dealing with a graspable, edible object that follows the article "the" and is a noun. The pseudo code of the repair method looks as follows:

- 1: unknownWord \leftarrow UnknownWord?(problem)
- 2: **if** unknownWord **then**
- 3: GenericConstruction \leftarrow CreateGenericConstruction(unknownWord)
- 4: Add(GenericConstruction, ConstructionInventory(node))
- 5: RestartSearchTree()
- 6: **else return** false

Yet, this repair is a smaller guarantee for communicative success than the previous one. When the parsing process succeeds after the repair has restarted it, the thereby extracted meaning can be interpreted in multiple ways. Any object on the table that can function as something that can be passed on to someone, is edible and that is out of the father's reach (assuming that this is not a silly language test) is eligible by the student. Of course, after the feedback received by the father on the real meaning of salad, another DR-pair might trigger to resolve the meaning feature in the generic construction.

5.3 FCG for self- and other-repair

Advanced language users often resort to the meta-level during linguistic processing. Their flexibility strategies function to secure that a linguistic interaction does not fail due to conversational slips or inaccuracies. Or as Postma and Kolk (1993, p. 472) put it: "Self-repairing of speech errors demonstrates that speakers possess a monitoring device with which they verify the correctness of the speech flow". This linguistic monitoring is equal to what I have previously described as diagnostics and there is ample evidence that it does not only happen after errors have been articulated but also prior to its motoric execution, speakers can detect errors. Also in comprehension, similar diagnostics are

active that can repair misunderstandings early and repair them internally or externally (by correcting the interlocutor's mistake explicitly).

This natural process of self and other repair occurs frequently in natural language conversations. To reach their joint goal in a conversation and minimize collaborative effort, speakers make *self-repairs* as soon as they detect a problem, as in the following exchange between two friends:

Example 2

- A: is it . how much does Norman get off –
- B: pardon
- A: how much does Norman get off
- B: oh, only Friday and Monday
- A: m
- B: [continues] ()

Other-repairs are often related to error correction, where an error is defined as a deviation from the norm that is handled in a speech community. Such deviations can be due to linguistic innovations or to insufficient grammatical knowledge, as we notice in children or adult second language learners. Parents correct their children more than is usually assumed and it is – although not exclusively – through their feedback that children are learning. An example of such other-repair shows how a mother corrects her son after she detected an incorrect past tense conjugation:

Example 3

- Child: I seed him.
- Mother: Ah, you saw him.
- Child: Yes, I saw him. (Corder, 1974, p. 25-26)

In language production self-repairs are often triggered by a failure during *re-entrance*, which is the process of parsing the utterance that a speaker has produced before it is verbalized (Steels, 2003). This is a standard goal test in FCG that competent language agents all share. A diagnostic that is specialized for re-entrance (which can serve as a learning situation) can then identify possible problems to lead to a misunderstanding of or an effortful processing of the intended utterance. Repairs that are triggered by these problems can then try to solve them so that a corrected utterance can be delivered. Sometimes, it is also the FCG constructions themselves that are responsible for self-repair. Inside an FCG search tree, failed construction application automatically leads to backtracking operations. When alternative search paths are possible, they will be explored until a potential solution is found. Figure 5.5 shows an example of such self-repair in parsing, where the misunderstanding happened early in the search tree but could be restored through backtracking.

Self-repair through re-entrance is only one example of how FCG processing can be made robust. Another possible way to ensure robustness lies in the manipulation of the regular construction application mechanisms (match and merge, see Section 3.2.1). In this

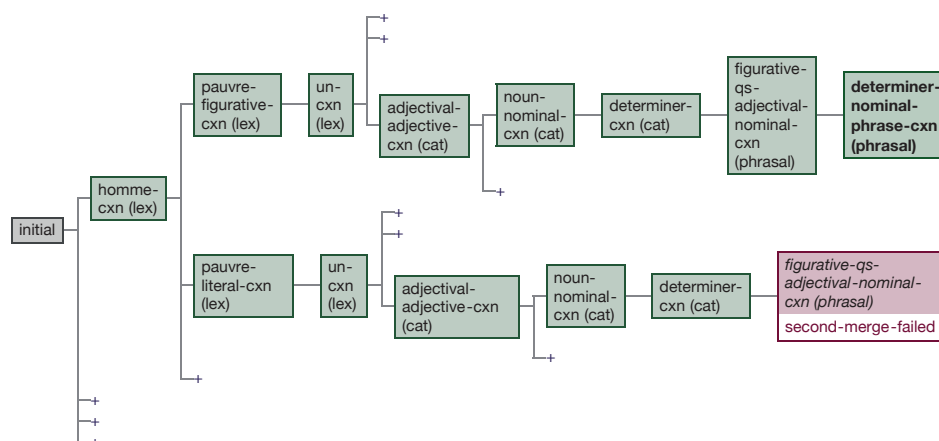


Figure 5.5 – Instead of signalling an error after the phrasal construction could not be merged with the existing transient structure, the FCG engine backtracks until a valid new search branch is detected, which here eventually leads to the correct solution. The difference between parsing “un pauvre homme” and “un homme pauvre” lies in its figurative versus literal meaning. This figure has been reprinted with permission from Bleys et al. (2011).

way, one can allow for new features to be merged into the transient structure. Instead of first having a matching phase, “FCG tests whether a construction can impose its feature structure through the merge operation” (Steels & van Trijp, 2011, p. 325). This operation requires only one parameter change in the grammar engine’s construction application options. The `merge-always` boolean is then by a repair temporarily set to true instead of its default NIL value.

In sum, the real value of flexibility strategies is that they *ensure robust language processing*, a property that is highly valuable for Intelligent Tutoring Systems research. It is only through *robust parsing methods* that a structural interpretation of the learner’s input can be rendered even in the presence of unexpected constructions. According to Nerbonne (2003), recognizing, diagnosing and reacting to student errors is a central technical issue in NLP. Yet, error classification is often claimed to be too difficult for existing technologies: when there are multiple parsing solutions and erroneous input might make sense in some of them. Even when it is clear that the input contains an error, it remains difficult to classify it. Most existing approaches either add rules to the grammar that explicitly cover the errors that learners typically make (Weischedel et al., 1978) or use a constraint relaxation technique (Murphy, Krüger, & Grieszl, 1998). Flexibility strategies offer an alternative solution to these.

Flexibility strategies offer an alternative solution to existing error analysis approaches that add error-rules to the grammar or use constraint relaxation techniques.

Box 5.5 – Flexibility strategies for other-repair

5.4 Conclusion

Meta-level computation is not only an important concept in programming languages but also plays a crucial role in language processing. As language users, we correct ourselves and others all the time by rephrasing our own utterances or repeating someone else's words. A meta-level computation component becomes thus indispensable when building a functional model of language processing. Construction application as it happens in the grammar engine should not only handle constructions by matching and merging (Section 3.2.1) but it needs a capacity to reason *about* this application process itself. This capacity is provided by an additional processing layer presented in this chapter in which processing problems are handled by *diagnostics* that identify them and *repairs* that test the outcome of certain solutions before regular grammar engine processing is restarted.

A flexibility strategy can be defined as a set of n diagnostics and m repairs that target one particular *processing problem*. Diagnostics check whether certain features are present in the transient linguistic structure or verify the results of the matching or merging operations (e.g. first/second merge failed). Every diagnostic is able to identify one type of processing problem. The identification happens as soon as the necessary conditions for the presence of this problem have been met. A repair is triggered as soon as a diagnostic signals a problem that the repair can solve. The repair will then proceed according to a predefined protocol in which each step tests a (sub)solution. Repairing can fail or succeed depending on the validity of the solution(s). In case of success, a repair can either decide to restart constructional processing or wait for another repair to come into action.

The concepts resulting from this chapter form an important contribution to the research of automated error analysis, a subfield of Computer-Assisted Language Learning that forms the heart of any intelligent language tutor (Heift & Schulze, 2007). Flexibility strategies are a useful format to capture typical learner problems and ways to solve them. The following chapter demonstrates how a language agent for Spanish verbs employs target-specific flexibility strategies to parse and restore L2 verb errors, made by English learners of Spanish. Finally, the same meta-level architecture returns in the design of learning strategies and tutoring strategies further in this dissertation.

Chapter 6

A language agent for Spanish verbs

With the FCG constructions and grammar engine settings needed to operationalize a Spanish grammar for verb conjugation and a general meta-level architecture that allows for flexible and robust parsing of learner errors, the language agent is gradually taking its form. As the previous chapter has shown, a competent language agent must be capable of identifying any minor error in the utterances that he is parsing. When the error can be restored to its originally intended form, linguistic processing can proceed without any visible interruptions. Yet, the error information can potentially also be used to explicitly offer a correction, which can in turn be used by a student agent to repair the original source of the error. This chapter shows the flexibility strategies that are needed for the case study of Spanish verb conjugation. Moreover, it also shows how a large construction inventory can be bootstrapped so that the language agent can produce the full verb conjugation of any Spanish infinitive that you give it.

I designed a set of flexibility strategies to detect and correct Spanish verb form errors without making use of contextual information (subject, temporal adverbs, etc.). To evaluate their effectiveness I extracted verb conjugation errors from a subcorpus of the Spanish Learner Language Oral Corpora (SPLLOC) (Mitchell, Dominguez, Maria, Myles, & Marsden, 2008) that specializes on the acquisition of tense and aspect. The language agent is then asked to parse these errors and return the best possible correction for them. Because the learner corpus is annotated with human tutor corrections for the hand coded verb form errors, the agent's correction can immediately be compared with the human's. Although no contextual information was used to assist the error correction process, the flexibility strategies reach an accuracy of 78%.

Some researchers in the field of Intelligent Computer-Assisted Language Learning (ICALL) have already considered the value of using Construction Grammar for modelling linguistic knowledge, both for the expert as for the student model. According to Matthews (1993)

Construction Grammar meets the three "criteria of adequacy" for intelligent language tutoring: (i) computational effectiveness (successful computational implementation), (ii) linguistic perspicuity (descriptive power) and (iii) acquisitional perspicuity (contribution to explanation of L2 acquisition). Also Schulze and Penner (2008, p. 427) already discussed the potential of CG "to overcome some challenges in ICALL and to facilitate a more thorough analysis of learner language in context and thus improve our knowledge about language learning processes". This in-depth analysis of learner errors forms the topic of the current chapter.

Construction Grammar meets the three "criteria of adequacy" for intelligent language tutoring.

Box 6.1 – The potential of Construction Grammar for ICALL

Section 6.1 first shows how the language agent's Spanish grammar was bootstrapped by using a decision tree from a previous research to build an online Spanish verb conjugator. Second, Section 6.2 presents the diagnostics and repairs that are implemented to support all flexibility strategies needed to capture typical learner errors for Spanish verbs that occur in a Spanish learner corpus (SPLLOC). Finally, Section 6.3 tests the flexibility strategies on verb errors taken from the SPLLOC data set.

6.1 Bootstrapping a Spanish verb grammar

Yet, because developing a full language learning strategy for Spanish verbs is a time-consuming task unless diagnostics and repairs from a related language can be reused and adapted, I will first look here at an alternative approach to creating a life-size grammar that can process all Spanish verb forms. In such an approach a grammar is created by algorithms that automatically generate all constructions that are required by a particular verbal paradigm. These algorithms start from a classification of the infinitive that is fed to the system. According to such a classification (which is done by a decision tree), every verb in the Spanish grammar (and even neologisms) can be conjugated once its infinitive is known. The current section contains details on how to implement such an automatic grammar creator. Section ?? shows how the same target grammar can be learned incrementally through the use of meta-level learning strategies.

A fast way to supply a basic language agent with a large-scale grammar is to create an algorithm that adds the constructions needed to conjugate a verb based on its infinitive. Obviously, the language agent's construction inventory does not only consist of a large number of verbs but also contains their appropriate conjugational rules. When a verb such as *pensar* 'to think' is added to the construction inventory, it should be marked that

it follows the $e \rightarrow ie$ diphthongization process when the stem vowel receives stress. Section 6.1.1 describes the automatic classification of verbs according to their conjugation paradigm. Section 6.1.2 shows how this decision tree is used to construct a grammar for the 564 most commonly used verbs in Spanish (worldwide).

6.1.1 A decision tree for automatic grammar creation

To add a new verb to his construction inventory an agent typically has to go through a complex process that not only creates a lexical construction for the verb stem, but also adds the morpho-phonological constructions that are required to cover the full verb paradigm. This process can be bootstrapped so that we can automatically create a grammar that can conjugate a whole range of verb paradigms belonging to a list of infinitives that is given. To launch this automatically generated process of construction creation, a classification task needs to be solved, which defines the verb type of the infinitive and along with that provides details on how to conjugate it.

Rello and Basterrechea (2011) have investigated a similar conjugation task in the development of *Onoma*, a verb conjugator that provides the conjugational paradigm for any infinitive that you type in (given that you place the diacritic accents correctly), including neologisms. Their application can be tried out online at www.onoma.es. *Onoma* also analyzes a verb form that you give it: e.g. "saqué" is analyzed as a first person singular past perfect form of the verb *sacar* 'to gain, to receive'.

The *Onoma* system is made up of two modules: a classifier and a modeling module. The former "is designed to recognize the verb form and extract the information needed for its conjugation or analysis" (Rello & Basterrechea, 2011, p. 230). The FCG classifier also contains a submodule to identify the stress pattern of the verb. Verbs are then classified into (a) regular, (b) irregular and (c) semi-regular verbs. When identified, irregular verbs are further divided into (b1) the so-called *primary verbs* and their derivations: *traer* 'to bring', *valer* 'to be worth', *salir* 'to go out', *tener* 'to have', *venir* 'to come', *poner* 'to put', *hacer* 'to do', *decir* 'to say', *poder* 'can', *querer* 'to want', *saber* 'to know', *caber* 'to fit', *andar* 'to walk'; and (b2) the irreducible verbs, a set of six verbs whose conjugational paradigm is entirely stored in memory: the auxiliary verb *haber* 'to have', the copulative verbs *ser* 'to be' (permanently) and *estar* 'to be' (temporary), and the monosyllabic verbs *ir* 'to go', *dar* 'to give' and *ver* 'to see'. The semi-regulars are further split up into (c1) verbs that engage in diphthongization or a vowel displacement in their root; (c2) verbs which are affected by irregular diacritic rules and (c3) verbs whose root ends on a vowel and that undergo heterogeneous rules of irregularity. As stated by the *Onoma* engineers (Rello & Basterrechea, 2011, p. 230), "apart from the irreducible verbs, the rest of the verbal paradigm system is based entirely on rules and patterns implemented in Module 2".

The modeling module implementation differs between Onoma and FCG. Since Onoma does not store constructions in memory that apply in parsing and production but creates a full conjugational paradigm when a user enters an infinitive, the system first builds hypothetical verb forms by means of several finite state machines. This verb form is then modified in a subsequent step. The hypothetical verb form in FCG can be mapped to a lexical construction for a verb stem. Hypothetical endings are not modeled by this FCG grammar, as verb endings are selected based on the classification of the verb. The modification of the stem happens by so-called phonological constructions (see Section 4.2.4). We follow the same irregularity rules and patterns as those defined in the Onoma manual (Basterrechea & Rello, 2010):

- Pattern **To**: verbs whose root contains the stressed syllable
- Pattern **Te**: verbs whose ending contains the stressed syllable
- Pattern **Dei**: verbs whose endings begin with *e* or *i*
- Pattern **Dao**: verbs whose endings begin with *a* or *o*
- Pattern **Di**: verbs with a stressed ending that begins with an unstressed *i*
- Pattern **Dti**: verbs whose endings begin with a stressed *i*
- Pattern **Dt-i**: verbs with a stressed ending that begins with any vowel except *i*

There are also two additional patterns for the *primary verb* conjugation:

- Pattern **Fc**: all person forms of future and conditional tenses/indicative moods.
- Pattern **í4**: all person forms belonging to the preterite perfect simple tense/indicative, the preterite imperfect/subjunctive and the future/subjunctive.

Depending on these patterns and the formal composition of the verb form itself, a specific irregularity rule is activated that modifies the hypothetical verb form. Overall, Onoma contains 40 irregularity rules that are divided into five groups. The FCG conjugator uses only 30 irregularity rules (Table 6.1), also divided into the following five groups (Rello & Basterrechea, 2011, p. 232):

1. Consonantal orthographic transduction rules: these modify the verb in order to ensure that the derived form obeys Spanish orthographic conventions. For instance, *sigo* 'I follow' is derived from *seguir* (stem *segu-*) 'to follow', when activated by the **Dao** pattern.

Table 6.1 – The Onoma conjugator uses 10 additional irregularity rules compared to the number of FCG constructions. Every row shows the number of rules per system for the five rule groups that have been identified by Rello & Basterrechea (Rello & Basterrechea, 2011, p. 232). The difference in rule number is especially visible in groups 4 and 5, that is vowel root ending rules and specific primary verb rules.

	Onoma	FCG
1	9	9
2	2	2
3	8	11
4	8	6
5	13	2
sum	40	30

2. Diacritic transduction rules: these are activated by the **To** pattern, as in *vacío* 'I empty' from *vaciar* 'to empty'.
3. Root vowel transduction rules: these are activated by patterns **To** and **Dti** and operate on the root vowel, which can either be diphthongized or replaced by another vowel. E.g. *sirvo* 'I serve' from *servir* 'to serve'.
4. Vowel root ending transduction rules: these rules affect verbs whose infinitive form root ends in a vowel: e.g. *oyes* 'you hear' has been derived from *oír* 'to hear' by additional of the letter *y* after the root (activated by pattern **Te**).
5. Specific primary verbs transduction rules: these rules are activated by the patterns **Fc**, **í4**, **Dao** and **To**. E.g. *tuve* 'he/she had' is changed by activation of pattern **í4**; *tengo* 'I have' is modified by adding the letter *g* after its root (activated by **Dao**) and *tendré* 'I will have' is altered by addition of a *d* after the root in the verb forms recognized by pattern **Fc**.

In Onoma, these rules are consulted every time a user requests the conjugation of a particular infinitive (which can be a non existing verb as well). In FCG, once a verb of a particular "family" is added to the grammar together with the phonological and morphological constructions that define its final form, any future construction that belongs to the same family is spared from this grammar-enlarging procedure. Moreover, the grammar remains stable once a particular verb has been recorded. The language processing engine will cycle through the grammar and apply the appropriate constructions at the appropriate moments in the processing pipeline.

When a new verb is conjugated in Onoma or added to the grammar in FCG, it is classified according to a decision tree that determines whether it is irregular and if so, according

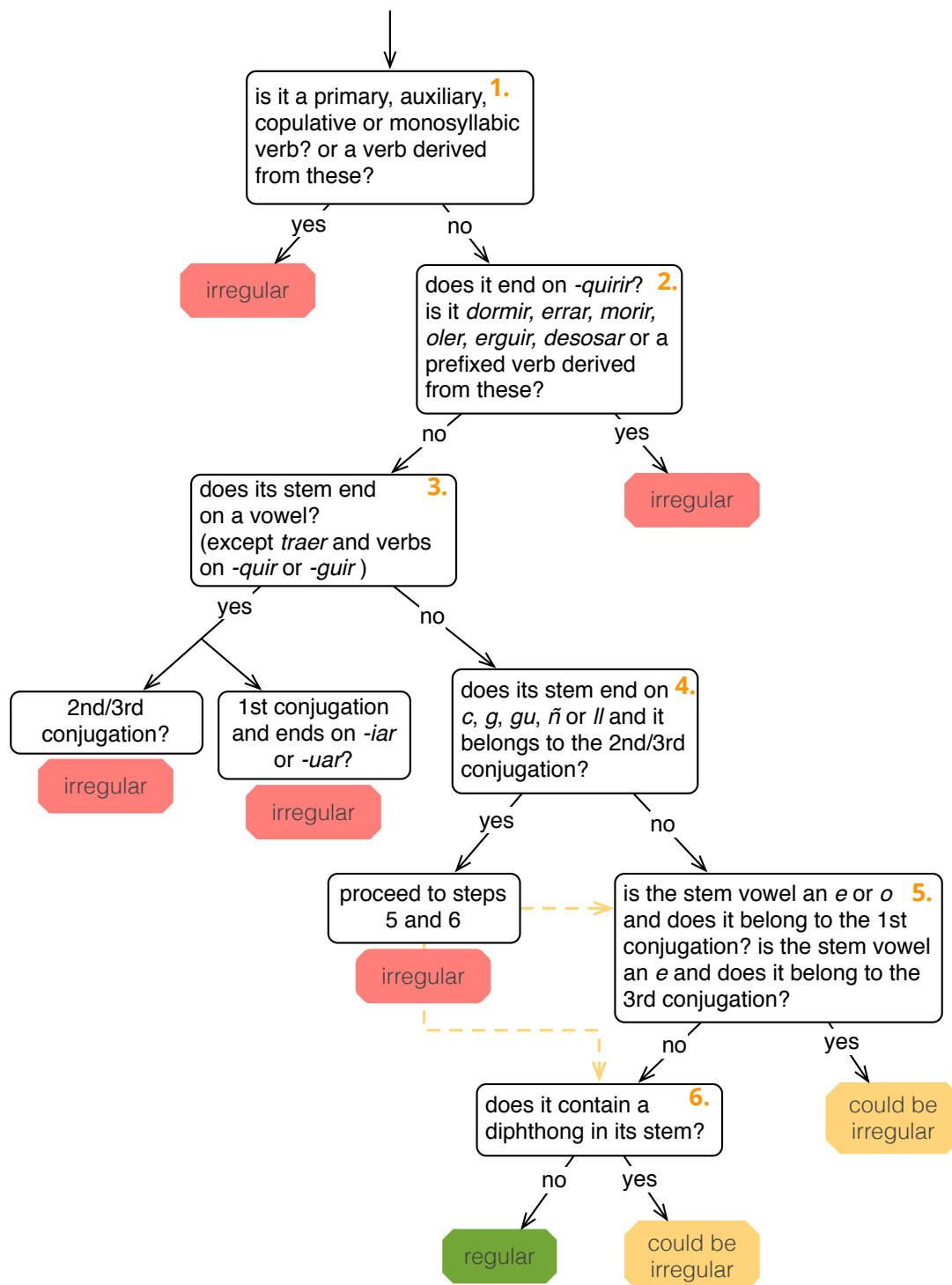


Figure 6.1 – The decision tree consists of six main splits to define whether an infinitive is regular or irregular. Most of the nodes perform checks on the verb stem's last letters and its main vowel. When an infinitive passes through the complete tree with only negative answers it is found to be regular.

to which irregularity rule it can be conjugated (Figure 6.1). In FCG, irregularity rules correspond to construction families: *herir*-verbs, *errar*-verbs, *jugar*-verbs, etc. Some steps in the decision tree trigger further decisions, such as question 4 "does its stem end on *c*, *g*, *gu*, *ñ*, *ll* and it belongs to the 2nd/3rd conjugation?" which can be followed by steps 5 or 6. Examples of infinitives that are characterized by two construction families are *rogar* 'to request' and *colgar* 'to hang up' (families *delegar* and *contar*); *conseguir* 'to manage', conjugated according to verb families *distinguir* (group 4) and *servir* (group 5) and *comenzar* 'to begin', which belongs to the *trazar* (group 4) and *pensar* (group 5) verb families.

6.1.2 Conjugating Spanish verb forms

With the conjugation decision tree implemented as a cascade of functions in Common Lisp that can be called by the FCG engine, we can now start creating large-size grammars that can be provided to language agents in the form of construction inventories that boost their competence level. Yet, before we start classifying verbs and creating constructions that can cover their conjugation paradigms, we need a set of basic constructions that is used by any verb that is conjugated. This set was created by hand. It contains the following constructions, which are grouped according to their function:

- Agreement constructions (phrasal): add agreement meaning
- Tense-aspect-mood constructions (phrasal): add tam meaning
- Morphological constructions: person/number endings and tam endings (including participles, gerunds) and a reflexive construction
- (Morpho-)Phonological constructions: accent changes (*a*→*á*, *e*→*é*, etc.), assimilations (*i+is* →*ís*) and two stem placement constructions (assign stress pattern to the stem)

The base grammar contains a total of 68 constructions, of which 35 are morphological constructions, 21 phrasal constructions (agreement and tense-aspect-mood) and only 12 morpho-phonological constructions (see Figure 6.2a). No single lexical construction is present in this base grammar. However, as soon as the first verb is added, additional constructions are created for the auxiliary verb *haber*, 'to have'. Because *haber* is an irregular verb, the irregular forms in its conjugation are added as separate lexical constructions, which would bring the total of the base grammar to 105 constructions.

Jehle grammar

To create a large scale grammar on which the classification algorithm could be tested, I used a corpus that I received from Fred Jehle (University of New Mexico) with 11467 conjugated verb forms, together with their infinitives. The conjugated forms in the corpus represent roughly the 600 most common verbs in Spanish. It was used on <http://users.ipfw.edu/jehle/VERBLIST.HTM> to present verb paradigms to students of Spanish, who could use it to look up an infinitive and retrieve its full conjugation.

Once all the infinitives from the Jehle corpus have been added to the FCG base grammar and the classification has been done, the resulting grammar has a size of 1575 constructions. Lexical constructions are by far the majority in this construction inventory with a share of 93% (1466 constructions), followed by phrasal constructions (21), morphological (35) and morpho-phonological constructions (53). There were thus no new phrasal or morphological constructions added in the grammar expansion phase. The classification tree only creates new lexical and morpho-phonological (phon) constructions. The high amount of lexical constructions is due to the number of irregular verbs that were conjugated, which resulted all together in 866 constructions, while only 564 infinitives were added to the grammar.

The majority of the 564 infinitives (58%) are regular verbs that were not classified according to a particular verb type (see Figure 6.2b) and thus passed through the complete decision tree. The remaining 42% are irregular and semi-regular verbs that do not deviate from the regular verb conjugation paradigm on a number of verb forms. A total of 25 semi-regular verb types could be detected in the corpus data, with the most frequent verb type *secar* occurring 39 times. Seven of the verb types only had a single infinitive that was conjugated according to the type. The verb type frequency distribution is included in Figure 6.2c.

Evaluation

One way to verify whether the automatic verb classification is successful, is to parse a given verb form and reproduce it. When the result is the same as the original verb form, you know that the conjugation was correct. The evaluation tested 33954 verb forms, belonging to 566 verbs, and yielded on average 18 errors (5 runs). All these errors were due to ambiguous parses of verb forms that have a stem that can belong to any of two infinitives: *sentir/sentar*, *regir/regar*, *presentir/presentar*, *sentir/sentar*. Figure 6.3 contains an example of such an ambiguous parsing process where the verb stem *present-* can lead to *presentar* 'to present' or *presentir* 'to anticipate, to sense'. Because the testing function only takes one solution into account, one of the two lexical constructions that cover the stem *present-* triggers randomly. A solution to this ambiguity problem would be to implement a node test that checks whether the parsed meaning so far is still

6.1. BOOTSTRAPPING A SPANISH VERB GRAMMAR

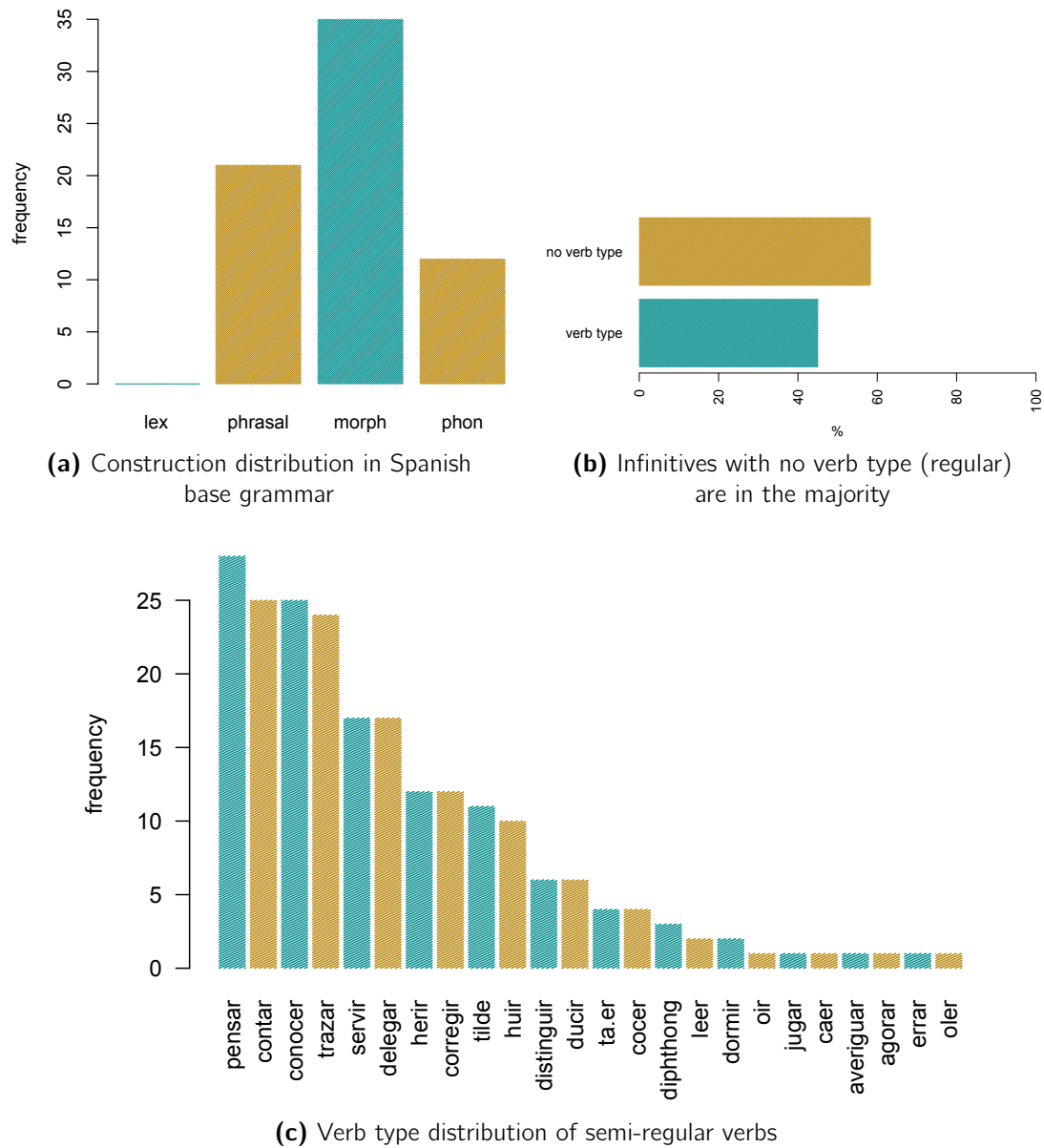


Figure 6.2 – General statistics of the automatically generated construction inventory

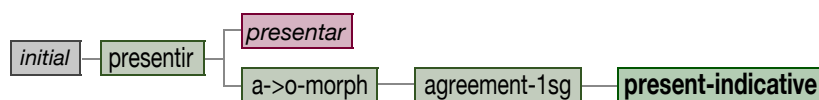


Figure 6.3 – Parsing “presento” leads here to the wrong solution, with *presentir* as the main verb (stem “present”) instead of *presentar*. When the parsed meaning is reproduced a different verb form is uttered: “presiento”.

compatible with the situation that is being observed (the meaning). In a real language game, this checking is done automatically but in this verb form testing function, only the decontextualized verb form is available.

Discussion

Conjugating the full verb form corpus with around 34 000 (x2) parsing and production processes is an expensive operation in FCG. Construction application slows down due to the large size of the lexical construction set (1466 constructions). Because FCG’s search process to find the next construction is by default completely blind, on average 733 constructions (50%) have to be matched to the transient linguistic structure before the correct one is found. Apart from general heuristics that optimize the search process, I converted the largest construction set (lexical) into a *hashed construction set*. Different from a regular construction set, it constructs a hash table for the form strings and the meaning predicates of all lexical constructions, which can be used to quickly retrieve the constructions that are relevant for meaning that is being produced or the forms that are parsed. Hashed construction sets have now become standard in FCG since my first introduction.

Although the automatic generation of a large scale grammar solely based on infinitives is a fast way to create life-size grammars, the decision tree that is used to classify verbs and create the constructions that they demand accordingly is not a realistic representation of a human learning strategy. Instead, with the right learning strategy a language agent can acquire a construction inventory. Assuming that the learning language agent is aware of the general linguistic patterns of the Spanish language, the next section shows how a set of manually coded diagnostics and repairs could encode a one-shot learning tactic that is able to reach a full competence of the target language system.

6.2 Diagnostics and repairs for verb form processing

How can a verb form error be detected by a language agent? As learners typically make the same kinds of mistakes, a list of problems can be created that the language agent

might encounter during the parsing of learners' utterances. This section presents a set of flexibility strategies that can catch these errors, correct them and return the type of error that has been committed. The most commonly encountered errors are the following:

- **Verb class:** the inflectional ending belongs to a different verb class than the verb stem, e.g. *asustió (correction: asustó, he/she scared);
- **Tense, aspect, and mood:** the verb form is conjugated correctly but carries the wrong tense, aspect or mood, e.g. indicative instead of subjunctive;
- **Person, number** (agreement): the verb form is conjugated correctly but carries the wrong person or number feature, e.g. 1st person singular instead of 1st person plural;
- **Phonological:** the verb form is conjugated through regular patterns but should have undergone phonological changes such as assimilation, palatalization, fronting, etc., e.g. *juga (correction: juega, he/she plays); or
- **Verb stem:** the verb stem is not part of the lexical inventory of the Spanish language due to errors in spelling, pronunciation or missing lexical knowledge, e.g. *descubir (correction: descubrir, to discover).

Of course, any combination of these errors is also likely to occur. For instance, a wrong verb form that is built on a non-existing verb stem and the wrong verb class such as in *descub-aba, where the correct stem would have been descubr- and the correct ending for the past imperfect 3sg form -ía, resulting in "descubría" (he/she discovered). Figure 6.4 collects all diagnostics and repairs that a Spanish language agent needs for detecting and correcting learner errors. They are presented in more details in the following sections. All diagnostics and repairs included here operate exclusively in parsing.

6.2.1 Feature mismatch

A feature mismatch occurs during the merging operation of a construction that succeeded the FCG matching phase onto an existing transient linguistic structure that contains incompatible feature values. A realistic example would be the erroneous verb form "jugía" (3sg past imperfect of *jugar* 'to play'). The diagnostic that detects feature mismatches comes into action as soon as a **second-merge-failed** status is returned by a merging operation. In the case of *jugi'a*, this status is evoked when the -ía morph construction for the suffix could not be integrated into the transient linguistic structure that already contains the verb stem *jug*. One alternative construction could apply after this merge failure: the default (unmarked) construction for 1sg/3sg endings in the present indicative.

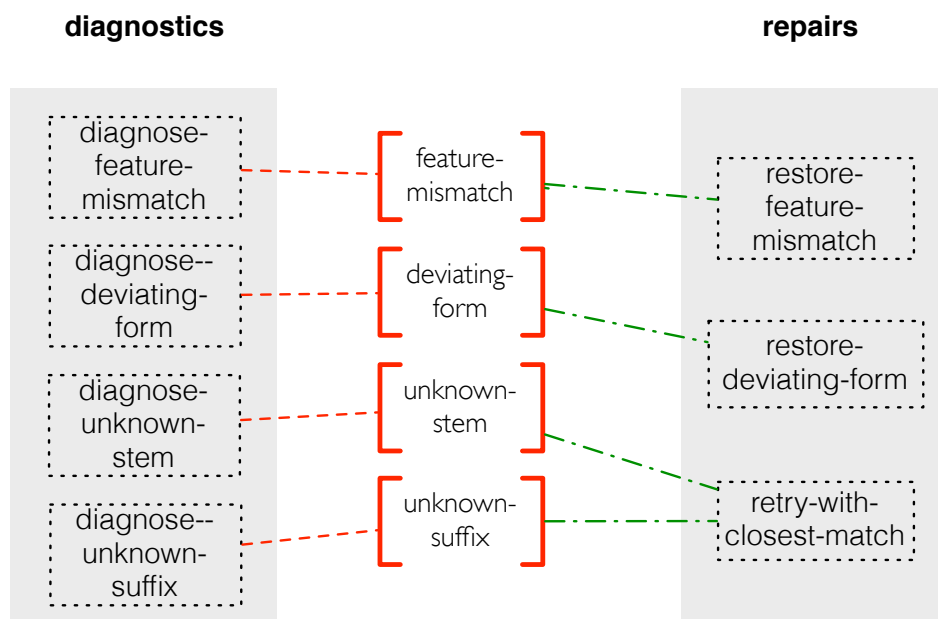


Figure 6.4 – A language agent uses four flexibility strategies to diagnose and repair ungrammatical utterances.

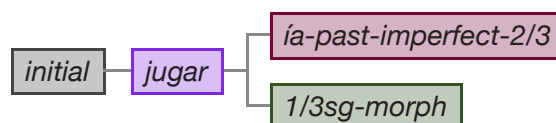


Figure 6.5 – The parse tree of “jugía” (he/she played) is interrupted after parsing the second/ third verb class ending “ía” that does not match the first verb class stem. The default 1/3sg morphological construction can apply and yields a present tense reading, which does not fit the intended meaning (past imperfect).

Nevertheless, the “i’a” suffix string is left unprocessed in the top unit of the FCG transient structure. Figure 6.5 contains the resulting FCG parsing application process.

Given this particular problem of a feature value mismatch, a repair will further pinpoint the causes of the mismatch and return a possible correction that restores this error. By accessing the merge results, the dissimilar feature value is revealed (through a comparison of the functional pole of the *-ía* construction and the transient structure’s syntactic pole): the verb class feature. The verb stem’s verb class feature in the transient structure ((1 +) (2/3 – – –)) (*jugar* belongs to the first verb class) did not match the feature matrix of the suffix: (==1 (1 –) (2/3 + ?2 ?3)). *Jugar* needs a suffix of the first verb class, not one that combines with stems of the 2nd or the 3rd verb class. The repair returns this information together with the corrected verb form (stem + ending). The ending could be retrieved from the FCG grammar since the diagnostic could tell use the correct verb class, tense, aspect and agreement features.

6.2.2 Deviating form

The deviating form problem is detected by a diagnostic that verifies whether the language agent that parsed a particular verb form would have said the same when he were the speaker. This comparison can be done by reproducing the parsed meaning. Similar to re-entrance, this hidden production process simulates the original student agent's processes. Reproduction is only run when a fully compatible meaning was parsed (containing all expected meaning predicates) to avoid irrelevant reproductions. An example is the parsing of the verb form "juga" (3sg present indicative of *jugar* 'to play'), which results in the following meaning:

```
((3sg-agent ?agent ?context) (jugar ?event ?context)
 (time-sphere ?event ?time-point present) (event-overlaps ?time-point)).
```

This meaning is then re-produced by the language agent and results in a different verb form: "juega" 'he plays'. The string difference between *juga* and *juega* is enough to trigger a deviating form problem. The repair that is specified for this problem runs through the following steps:

1. The dissimilarities between the two transient linguistic structures are compared by a unification operation on each linguistic feature (e.g. syn-cat, sem-cat, phon-cat, form, meaning, etc.).
2. The set of mismatched features is returned. In the *juga/juega* example this is the nucleus feature of the **phon-cat**.
3. The repair returns the corrected form *juega* and the stem nucleus feature that caused the error.

6.2.3 Unknown stem/suffix

When the previous two diagnostics did not trigger, one last processing problem that is often found is the presence of unprocessed strings at the end of a parsing process. An appropriate example is here the utterance "juqaba" (correction: "jugaba"). The stem "juq" is unknown to the FCG expert system so parsing stops immediately since a suffix alone cannot lead to a successful parse. Given that the segmentation algorithm segmented the verb form into a stem "juq" and an ending "aba" (1/3sg past imperfect), a diagnostic now signals an unknown stem. A related problem is the unknown suffix problem, where the stem can be parsed but the suffix is left unprocessed in the top unit.

An unknown stem repair will then try to solve this problem by using the language agent's constructional knowledge to find a verb stem in his construction inventory that most closely resembles the string that could not be parsed. This comparison is done by a simple Levenshtein distance metric that compares all stems of the language agent's grammar with the one present in the current problem. Because the first letter is usually less error prone than the rest of a word when typical learner errors are examined (Yannakoudakis & Fawthrop, 1983; Sibley, Pollock, & Zamora, 1984), I extended the measure to put additional weights on the first letter. After the string comparisons, the parsing process is restarted with the verb stem that scored highest on the similarity test. The new form is now the correction "jugaba" and parsing will succeed. The repair returns the corrected form and the fact that the verb stem the learner provided was unknown to the system. In a future implementation, the specific position in the stem that caused the failure (in this case "q" was used instead of "g") could be returned as well in order to provide more precise feedback.

6.2.4 Remaining errors

Remaining errors are either agreement errors (different person or number) or the use of a different tense, aspect or mood than the one that was expected. Such errors can only be tracked if there is a target meaning available to the expert system (i.e. exercises that have a well-defined solution). In such cases, a diagnostic to detect a meaning mismatch is appropriate. This FCG diagnostic compares the feature structure that results from parsing the learning utterance and the production result of the target meaning as soon as the parsed meaning differs from the target meaning. The mismatch in feature is then returned together with the corrected utterance. Combinations of multiple errors will automatically be captured as multiple diagnostics can trigger in a row and solve each error independently.

6.3 Corpus-based error analysis

The Spanish Language Learner Oral Corpus (SPLLOC) was used to evaluate the language agent's error analysis component. The SPLLOC is a second language learner corpus that contains exclusively oral data that has been collected and transcribed (CHILDES format) (MacWhinney, 1991) during the ESRC funded project "Linguistic development in L2 Spanish: creation and analysis of a learner corpus" (University of Southampton, University of Newcastle) (Mitchell et al., 2008). The material that has been collected includes learner narratives, interviews and picture description tasks. This material is freely

Table 6.2 – Levels in the SPLLOC2 corpus.

L2 Spanish level	Typical age	Number of hours of Spanish instruction	Educational level	Position on CEFR
low	14-15	ca. 240 hours	lower secondary school	A2
intermediate	17-18	ca. 750 hours	sixth form college	B1-B2
advanced	21-22	ca. 895 + year abroad	university	C1-C2

available for use (www.splloc.soton.ac.uk). The corpus can be consulted in audio format (Wave, MP 3) and as transcribed text (transcription file, tagged file, XML).

Section 6.3.1 describes the verb forms that were selected from the SPLLOC to carry out the evaluation. The results of this analysis are presented in Section 6.3.2.

6.3.1 The learner corpus

The SPLLOC 2 project is titled "The Emergence and Development of the Tense-Aspect System in L2 Spanish". The corpus contains samples of spoken Spanish produced by 60 instructed learners with English as their L1. The learners are all English L1 speakers who have learned L2 Spanish in educational contexts within the UK. Speakers from bilingual English/Spanish backgrounds or with extensive social contacts with Spanish speakers were excluded from the sample. According to the SPLLOC investigators, a learner selection based on gender was impossible because the large majority of L2 learners at college and university levels in the UK are female. Table 6.2 summarizes the levels present in the SPLLOC 2.

The learners all undertook five tasks. Tasks 1 to 4 were designed to explore the learner's knowledge of the past tense in Spanish, and to relate past events in sequences in multiple contexts, ranging from more controlled to open tasks (narrative tasks, guided interview, simultaneous events task). All activities were undertaken individually with a member of the research team. The fifth task was a computer-based interpretation task, and was specifically designed to explore learners' developing ability to distinguish different meanings of the Spanish imperfect and preterit. All task descriptions can be downloaded from the SPLLOC website.

For the evaluation of the FCG error analysis component I selected the lowest and the highest L2 learner level that was available in the SPLLOC 2 corpus: low intermediate (Lower secondary school) and advanced (University Year 4). For these two groups of 40 learners (20 per level), I used the four spoken tasks and extracted all errors automatically from the XML file, together with the native corrections. This results in 500 verb form

errors. The majority of errors come from the low intermediate group: 82% or 408 errors against 92 errors in the advanced learners subcorpus. This results in on average 20,4 errors/learner in the low intermediate group and 4,6 errors per learner in the advanced group. I include an example of an utterance that was extracted here below. The learner's error "venieron" is corrected here as "vinieron" *they came* (stem change).

```
<u who="C01" uID="u11">
  <w>después</w>
  <pause symbolic-length="simple"/>
  <g>
    <w>
      venieron
      <replacement>
        <w>vinieron</w>
      </replacement>
    </w>
    <error/>
  </g>
  <w>a</w>
  <w>su</w>
  <w>casa</w>
  <t type="p"/>
</u>
```

The language agent does not only need flexibility strategies to detect and correct the corpus errors but also a construction inventory and grammar engine that can parse and produce all corrected verb forms. This construction inventory is created by feeding all corresponding infinitives of the corpus errors to an automatic grammar creation algorithm that instantiates constructions for the verb's full paradigm. This algorithm is explained in Chapter 8. The 500 corpus error forms can be traced back to a total of 93 infinitives. The FCG grammar contains 490 constructions to operationalize all conjugations. The most frequently used infinitives are: "perseguir" (36), "leer" (34), "jugar" (24), "haber" (22) "tranquilizar" (21) and "despertar" (21).

6.3.2 Evaluation Results

The FCG engine is given one verb form at a time that needs to be parsed. If it cannot be parsed with the constructions that are part of the grammar, the meta-layer catches the form and searches for a solution that can be parsed (either by transforming the input or by modifications to the transient feature structure). The final parsed meaning is thus a hypothesis that the language agent makes about what was meant by the student when

he produced the verb form. This meaning is then reproduced by the language agent and returned as the corrected verb form. The source of the error can also be made explicit but is not relevant for the purposes of the current task.

The performance of the language agent's corrections is measured in terms of accuracy, which is defined as follows: Accuracy is the *percentage of corrected forms that are the same as the human correction* (provided by the corpus). Table 6.3 summarizes the accuracy scores for the two subcorpora. The average score on the complete subcorpus (low intermediate and advanced) equals 64%. Group scores differ considerably: 57.6% of accurate corrections in the low intermediate group against 70.7% in the advanced group. Apart from the fact that the advanced errors only constituted 12% of the errors that were investigated, this difference also suggests that the types of errors made by more advanced learners are easier to detect by the flexibility strategies.

Correction accuracy is the percentage of verb forms corrected by the language agent that are the same as the human correction provided by the corpus.

Box 6.2 – Correction accuracy

The second row in Table 6.3 includes the results of the so-called "enhanced accuracy". Sometimes FCG corrections are accurate when you only consider the isolated error form, without any notion of the discourse this form is embedded in. The correction of **haben* into *han* 'they have' (3pl indicative present of *haber*) is certainly valid although the human correction appears to be *hubo* 'he had' (3sg past anterior). Because the flexibility strategies do not have access to the context in which the verb error was made, such corrections can never be "accurate" according to the definition in Box 6.2. More examples of acceptable but strictly speaking inaccurate corrections are provided by Table 6.4. Sometimes the input form does not contain an error and can be parsed accordingly (e.g. "cantando"). Yet, because the human teacher knows what was meant by the student in this context ("cantando"), this form was marked as an error. The enhanced accuracy was calculated by going through all 500 error forms and the language agent's corrections

Table 6.3 – Advanced learners errors are easier to restore by the flexibility strategies, when pure accuracy is taken into account. The enhanced accuracy (corrections that differ from the original correction but are nevertheless appropriate) score reaches almost 80%.

	accuracy	enhanced accuracy
low intermediate	57.6	77.7
advanced	70.7	79.3
total	64.2	78.5

Table 6.4 – The language agent’s corrections are often closer to what an expert would predict without any knowledge about the context in which the verb form is used. Grammatical learner input forms are not corrected by the flexibility strategies.

learner input	language agent correction	human correction
contando	contando	cantando
habían	habían	había
*haben	han	hubo
*eban	eran	era
*pagá	paga	pagó
*preperé	preparé	preparó
*cogué	cogí	cogió
*elige	elige	eligí
*prepero	preparo	preparó
*cogue	coge	cogieron

by hand and judging them for their accurateness. In the accuracy picture that results from these counts 79% of the language agent’s corrections are accurate given the error verb form and no further contextual information. Moreover, the difference in accuracy has disappeared between the two levels.

Enhanced correction accuracy is the sum of the standard correction accuracy and the number of additional corrections that are acceptable given the lack of contextual information.

Box 6.3 – Enhanced correction accuracy

Figure 6.6a depicts the relative frequencies of the problems that were diagnosed by the language agent’s diagnostics: unknown stems are the most frequent problems (43% of all problems diagnosed), followed by verb class problems (feature mismatch) (26%) and stem changes (deviating form) (18%). Unknown suffixes (12%) and suffix changes due to assimilation (1%) complete the pie chart. Yet, as is illustrated by Figure 6.6, although unknown stem and unknown suffix problems are responsible for more than 50% of all errors, the language agent’s enhanced correction accuracy is lowest for these two error types. Errors related to changes in the stem can almost always be restored in a correct way by the language agent by the reproduction repair.

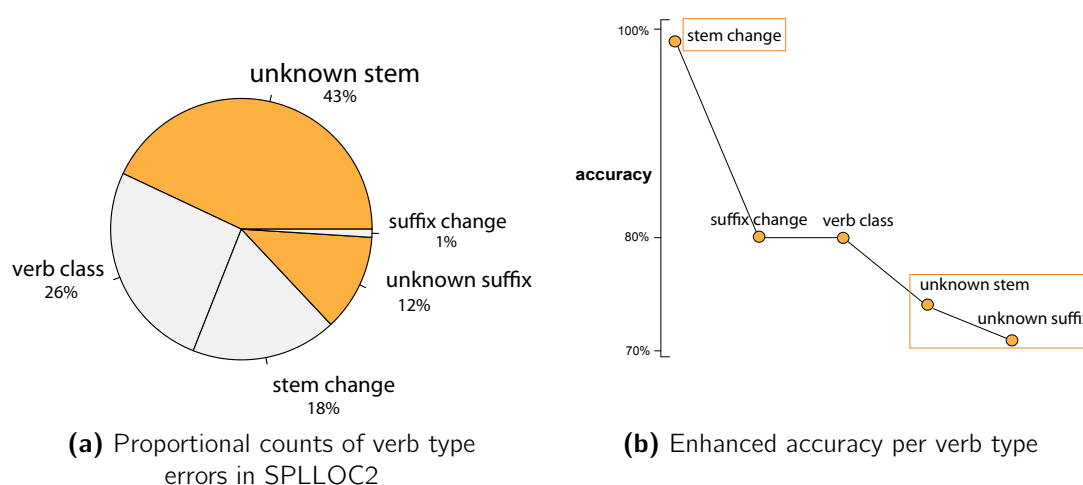


Figure 6.6 – The enhanced accuracy for the five error types is on average 77% (b). Stem change errors are easiest to correct, unknown stem and suffix errors the hardest. Yet, these errors form more than 50 of all errors present in the SPLLOC data set (a).

6.4 Conclusion

The first demonstration and evaluation of flexibility strategies on real-world data taken from a learner corpus has been carried out in this chapter. The processing problems that form the object of the flexibility strategies were formalized after a thorough analysis of the 500 error-coded verb forms that were extracted from the oral transcriptions of the SPLOC 2 data. The following errors occurred most frequently in the corpus: incorrect stem recall, incorrect suffix recall, mismatch in verb class between stem and suffix and inappropriate (or missing) morphophonemic stem changes. For each of the attested problems a flexibility strategy was designed, consisting of a set of generic operations that can be applied to any verb form in Spanish. The language agent is now complete, with:

- a construction inventory that contained constructions to conjugate all infinitives corresponding to the 500 corpus errors,
- a grammar engine that is optimized for verb form processing (segmentation, goal tests, etc.) and
- generic flexibility strategies that guarantee a flexible processing of ungrammatical utterances through the diagnostic/repair architecture that takes regular linguistic processing onto a meta-level where irregularities can be analysed and repaired.

CHAPTER 6. A LANGUAGE AGENT FOR SPANISH VERBS

The evaluation of this language agent when processing and correcting the 500 verb form errors made by learners of Spanish has yielded a basic correction accuracy of 64% and an enhanced accuracy score of 79%. Because error correction is carried out by a model of a proficient language user of the target language, the system can provide error type information "for free". Fluid Construction Grammar is now confirmed to be an adequate formalism for at least one component of the intelligent tutoring systems enterprise: error analysis and correction. The detailed information that can be gained from constructional processing will be made available for tutoring purposes in Part 3 of this dissertation.

Chapter 7

Learning strategies

If you have ever tried to learn a foreign language, you will have been confronted with various difficulties in acquiring certain domains of the language and potential "false friends" between your first language and the language you are learning (L2). How does this process work and why are some learners better or faster than others? According to Second Language Acquisition (SLA) specialists, "learners possess cognitive mechanisms which enable them to extract information about the L2 from the input" (Ellis, 1997, p. 5). Of course, adult learners bring an enormous amount of knowledge to the task of learning a new language:

1. They can draw on their first language to understand the meaning or use of certain words or grammatical constructions;
2. They have knowledge about the world;
3. They use communication strategies that help to make effective use of their L2 knowledge;
4. They know how language works in general (depends on the learner's language aptitude).

During learning, some linguistic features might have been fully mastered by the learner, while others have not yet been acquired. For instance, Ellis reports on a learner who accurately used the [progressive + to be] construction (e.g. *I am writing*) but had a very inaccurate knowledge of the past regular conjugation in English (**I writed*). Yet, studies suggest that learners acquire different aspects of a new language systematically, and that they follow particular developmental routes, with some features being learned faster than others (Ellis, 1997, p. 12).

Similar to L2 learners, a student agent has an incomplete construction inventory and grammar engine settings that have not yet been optimised for the target language system. Yet, with the same basic agent architecture as a language agent, which includes a construction inventory, a grammar engine and a meta-level architecture to capture problems, a student agent is capable of acquiring the language system of the competent language agent he is interacting with. The meta-level architecture is used to detect processing problems that arise due to insufficient linguistic knowledge (diagnostics) and solve them by following pre-specified steps that are part of the learning strategy (repairs).

The current chapter describes how meta-level computation is used to formalize such *learning strategies* that restore a student agent's failed production or parsing processes. Different from flexibility strategies, learning strategies actually modify the construction inventory every time a problem was repaired successfully either by expanding it with new constructions or by modifying existing ones. This type of meta-level learning is by definition *problem-driven*, rather than data-driven. Problem-driven learning, or also *one-shot learning*, typically happens very fast once a problem has been identified, and the learned construction(s) can be updated at a later stage when a new problem arises. Data-driven learning, on the other hand, proceeds much slower as a new construction can only be added when enough examples have been encountered.

Different from flexibility strategies, learning strategies actually modify the construction inventory every time a problem was repaired successfully either by expanding it with new constructions or by modifying existing ones.

Box 7.1 – Learning strategies versus flexibility strategies

The outline of the chapter is structured as follows: Section 7.1 explains the problem-driven learning approach by making the connection with Piaget's constructivism. Once the underlying theory has been covered, the rationale behind the use and the implementation of learning strategies will become clear in Section 7.2. Finally, a series of attested learning strategies in Second Language Acquisition are described in Section 7.3.

7.1 Problem-driven learning

The basic tenet of the constructivism learning theory is that humans are better able to understand and acquire a piece of information or a skill when they have construed it by themselves. In all constructivist theories, learning is seen as a social engagement that involves language, real world situations, interaction and collaboration among learners. During the learning process, the learner gathers information, converts it, formulates hypotheses and tests these hypotheses through interactions or experiences. Rather than

internalizing presented information as it is through rote-learning, the essence of learning lies in building constructs and internalizing the information that is given. Classrooms that follow a constructivist approach, must thus be transformed into a knowledge-construction site, where the teacher is seen as a facilitator and a guide. Students often "work in groups to approach problems and challenges in real world situations" (Ozer, 2004).

Jean Piaget and Lev Vygotsky (see Section 9.1.1) are two pioneers in the field of constructivist learning and have later also inspired computer-based learning environments such as the programming language Logo (Papert, 1980), ToonTalk (Kahn, 2004) and many others. Differences between Piaget and Vygotsky can be found in their ideas of how constructivism should be carried out in classrooms. For Piaget, the constructivist classroom must provide a variety of activities to challenge students "to accept individual differences, increase their readiness to learn, discover new ideas, and construct their own knowledge". Vygotsky, on the other hand, stresses the social aspect of learning more and promotes the active participation and collaboration of distinctive learners and the teacher: "the distance between the actual development of a child as determined by the independent problem solving, and the level of potential development as determined through problem solving under adult guidance or in collaboration with more peers" (Vygotsky, 1978).

Meta-level learning also follows a *constructivist learning perspective* because it is problem-based and the role of the language agent (or even more the tutor agent, see Part III) is to guide and challenge the learning process. The student agent learns about the target language system through the experience of problem solving. Problem-driven learning is fast as a single instance can suffice to learn something new. This type of learning relies greatly on the situation in which the learning instance occurs. It is also found in child language learning, and can be explained as follows: "typically, an infant who observes the consequence of a given action in a given context will readily be able to predict very well what happens if exactly the same action happens in the same context again" (Kaplan & Oudeyer, 2001, p. 35).

Although problem-driven learning is undoubtedly an important driver in the acquisition of a new language system, it is not the complete story of how learners perfect their linguistic skills and knowledge. We also use another type of learning to fully master a new language: data-driven learning. Different from problem-driven learning approaches, data-driven learning does not exclusively use local information that is situated in the current interaction but instead generalizes over a whole range of constructions and usage data. In fact, many of today's learning models follow a data-driven approach. Machine-learning techniques typically work on large amounts of data and learning happens through generalisation processes over this data. Learning in a data-driven approach occurs thus rather slowly, as many data instances must be processed before any type of generalization can occur. Patterns that are detected will be internalized so that future instances can be recognised faster.

The focus in this dissertation lies on problem-driven learning because it the most "traceable" type of learning, which is useful when using the student agent as a model of a real human student.

Box 7.2 – Problem-driven vs. data-driven learning

On the contrary, problem-driven learning is fast as a single instance can suffice to learn something new. This type of learning relies greatly on the situation in which the learning instance occurs. It is also found in child language learning, and can be explained as follows: "typically, an infant who observes the consequence of a given action in a given context will readily be able to predict very well what happens if exactly the same action happens in the same context again" (Kaplan & Oudeyer, 2001, p. 35). The same type of online learning - which must happen under the right circumstances, namely when the child is attending to the name object (Yu, L. Smith, & Pereira, 2008) - is also found in memory-based algorithms (Daelemans & Van den Bosch, 2005). We apply one-shot learning in our daily lives. When a new colleague is introduced to you in the office you will probably (unless you have problems remembering people's names) learn her name there on the spot, from a single example. The next time you want to address her, you can use her name straight away. Also, greetings in a foreign language are usually picked up in a single shot: hearing a Catalan speaker saying "adéu" when you leave the lift will often be enough to start using it yourself on future occasions (especially if you know French or Spanish, which use a similar strategy of saying goodbye by referring to God).

7.2 Meta-level learning strategies

Different from typical constructivist theories that focus on children's developmental processes, this dissertation focuses exclusively on *adult second language acquisition*. Learners construct utterances and come up with conceptualisations according to their personal learning strategies, which depend on their proficiency level and the learning objectives that they have set. If it is your goal to be communicative above all, you may start by using infinitives or only partially conjugating your verbs and compensate it through the use of adverbs (e.g. yesterday, until now, tomorrow) and deictic pronouns or gestures (e.g. "you" accompanied by pointing at the interlocutor). Or if you are lacking much of the vocabulary needed to talk about a certain topic, you might choose to rely on the general statistical patterns of the language you are learning together with words from your first language or a related language that you know. On the other hand, if it is your goal to reach a near-native language proficiency level, you might pay more attention to correct pronunciation or typical idioms used by native language users.

Learning strategies are here also implemented as meta-level operators, having exactly the same components as flexibility strategies (see Chapter 5). Diagnostics and repairs relate processing problems that appear not because of slips or innovative language use but because the construction inventory and grammar engine are not yet in a state to process utterances of the target language system without signalling a problem. As learning strategies are problem-driven, they typically make use of the information that can be extracted from the situation in which the problem is diagnosed as well as the constructions that were used to process the utterance or meaning so far. Because I am only considering second language acquisition here, I assume that the learner already has a basic knowledge of the linguistic patterns of the target language (e.g. word boundaries, morpheme composition, etc.).

Of course, learning strategies will depend on the type of language system you are focusing on (e.g. determiners, modal verbs, prepositions) and the specificity of the problems that are encountered. The following three prototypical examples of high-level diagnostics that are often used as part of a generic learning strategy:

- **New word in parsing:** The student agent detects a string in parsing for which he does (yet) not have a construction. Depending on the string's position and the agent's knowledge about other strings surrounding it, he can also diagnose the type of string that is unknown: e.g. lexical, morphological, functional, etc. A potential similarity measure could inform the language agent about overlap between the unknown word and the constructions already present in his construction inventory.
- **Unfamiliar word use:** This diagnostic can trigger in parsing when a word that can be covered by one of the student agent's constructions is for instance used with a different category than the one present in the grammar or follows a different word order pattern than the one expected.
- **Inexpressible meaning:** When the student agent's construction inventory is not yet mature enough, it can occur that he simply cannot express some meaning features that he would like to talk about. Moreover, the feeling of not being able to express oneself occurs frequently in language learners and even proficient language users encounter it sometimes. It is one of the main drivers for language innovation, especially when new concepts appear in society or multilingual speakers transfer expressions from one language into another one.

To really accomodate for fast learning that is problem-based repairs typically make use of construction templates that can be filled in, based on the features that are present in the problem. Example repairs for the **unknown-word** problem are described below. Other repairs include adding a new grammatical construction, modifying a word's phonetic properties, extending a word order pattern construction, etc.

- **Ready-made construction templates:** Construction templates contain most information for a particular construction such as a preliminary syntactic and semantic category (e.g. noun, object), word order information or phonetic constraints. When a new word of a certain type is detected, a new construction can easily be generated. When "ball" is detected as an unknown word but the learner is aware of its noun category given that he knows about the general linguistic patterns of the English language, he can add a construction by using the lexical skeleton template (which automatically adds the lexical category noun and the semantic type object):

```
(def-lex-cxn ball-cxn
  (def-lex-skeleton ball-cxn
    :meaning (== (ball ?x))
    :args ?x
    :string "ball"))
```

- **Elaborating constructions:** Later when the ball construction is encountered in new situations, it can be elaborated such as to provide information on plural formation, grammatical gender or typical collocations.

```
(def-lex-cat ball-cxn
  :sem-cat (==1 (graspable +))
  :syn-cat (==1 (plural regular)
            (gender common)))
```

When a repair is active in a particular branch of the search tree, a copy is made of the construction inventory and the grammar engine configurations. This copy is then modified by the repair and when the repair proves to be successful, the copy replaces the original construction inventory and configurations. A repair can be successful in two scenarios: (i) either the repair method could be executed until the end and thereby make its modifications as foreseen so that processing can continue, or (ii) a repair forces a restart of the processing in an earlier node of the search tree (it indicates also where) and success is defined upon the presence or absence of the previously diagnosed problem that the learning strategy tries to handle. When multiple problems are diagnosed in different branches, every branch hosts its own copy of the construction inventory. Only the final construction inventory most successful branch will be adopted as the student agent's new construction inventory.

Meta-level learning strategies are not used exclusively in problem-based learning approaches. Also more developmental learning styles can be modeled, given some predefined thresholds. The meta-level architecture has already been used to model first language acquisition that runs through three stages defined by Tomasello (2003): holophrases, item-based constructions and abstract constructions. Agents run through these phases,

first acquiring holophrases "gimme ball", then item-based constructions "gimme X", and finally abstract constructions `imperative + indirect object + direct object`. The transition from one stage to the next occurs when a learner has encountered enough instances of a certain type, which can be reflected in an absolute threshold or a success score of the constructions that are involved. Computational simulations of the acquisition of Russian aspect demonstrated this approach for a small set of Russian verbs (Gerasymova & Spranger, 2010; Gerasymova, Spranger, & Beuls, 2012). Hungarian subject-verb-object agreement was also subjected to the three-step developmental learning scheme (Beuls, Gerasymova, & van Trijp, 2010).

7.3 Attested learning strategies in SLA

Because we are interested here in second language acquisition (SLA), it is useful to investigate real learning strategies that have been observed in learners of a foreign language to incorporate some of the aspects of such strategies into the meta-level architecture of our student agent. The current section therefore discusses learning strategies that have been described in SLA research. It is meant to provide some background on how this term is usually understood and lists different types of learning strategies that have been verified. Readers who solely want to understand the architecture of the student agent can therefore easily skip this section.

Learning strategies, according to Weinstein and Mayer (1986), "have learning facilitation as a goal and are intentional on the part of the learner" (O'malley & Chamot, 1990, p. 43). For them, the goal of using a learning strategy is to "affect the learner's motivational or affective state, or the way in which the learner selects, acquires, organizes or integrates new knowledge" (Weinstein & Mayer, 1986, p. 315). This broad description gives rise to three classes of learning strategies, which have emerged from interviews with effective language learners: cognitive strategies, meta-cognitive strategies and social or affective strategies. They are discussed in the following sections.

7.3.1 Cognitive strategies

Cognitive strategies operate directly on information that is received by the learner during language processing. Weinstein and Mayer (1986) propose three broad groupings to distinguish these strategies: rehearsal, organization and elaboration processes. O'malley and Chamot (1990, p. 45) list the following typical cognitive strategies for listening and reading comprehension:

1. Rehearsal or repeating the names of items or objects that have been heard;

CHAPTER 7. LEARNING STRATEGIES

2. Organization or grouping and classifying words, terminology, or concepts according to their semantic or syntactic attributes;
3. Inferencing or using information in oral text to guess meanings of new linguistic items, predict outcomes or complete missing parts;
4. Summarizing or intermittently synthesizing what one has heard to ensure the information has been retained;
5. Deduction or applying rules to understand language;
6. Imagery or using visual images (either generated or actual) to understand and remember new verbal information;
7. Transfer or using known linguistic information to facilitate a new learning task; and
8. Elaboration - linking ideas contained in new information or integrating new ideas with known information.

The term learning strategies as I have used it in this chapter is most related to this first class of learning strategies. My interpretation of a learning strategy is much more tied to a particular linguistic phenomenon, which might constitute a hurdle for a learner. Yet, the general mechanisms described here as cognitive strategies are certainly also found in such language-specific learning strategies.

7.3.2 Meta-cognitive strategies

Meta-cognitive strategies are used by higher order executive skills such as planning, monitoring or evaluating the success of your learning task. O'malley and Chamot (1990, p. 44) provide the following examples for these strategies:

1. Selective attention for special aspects of a learning task, as in planning to listen for key words or phrases; Planning the organization of either written or spoken discourse;
2. Monitoring or reviewing attention to a task, monitoring comprehension for information that should be remembered or monitoring production while it is occurring; and
3. Evaluating or checking comprehension after completion of a receptive language activity, or evaluating language production after it has taken place.

These strategies are closer to the usage of flexibility strategies defined in Chapter 5 as they include re-entrance checks and evaluate comprehension at the end of a parsing process. If a certain irregularity is perceived, lower order cognitive strategies can be called to actually define the problem and solve it with the information that is at hand in the given learning situation.

7.3.3 Social/affective strategies

So-called social or affective strategies are those strategies that are involved either with interaction with another person (social) or that help to control the learner's feelings related to the learning task (affective). O'malley and Chamot (1990, p. 45) list the following strategies that are used by learners in listening comprehension:

1. Cooperation or working with peers to solve a problem, pool information, check notes or get feedback on a learning activity;
2. Questioning for clarification or eliciting from a teacher or peer additional explanation, rephrasing or examples; and
3. Self-talk or using mental control to assure oneself that a learning activity will be successful or to reduce anxiety about a task.

SLA learning strategies often rely on *conversational repair strategies* used by learners to gain a full understanding of a particular conversation. There are a few studies that report on the repair strategies that are practiced by students (Egbert, 1998; Liebscher, 2003). Egbert (Egbert, 1998) investigated the types of repair initiations that first year college German students employed in language proficiency interviews. The most common repairs are those that learners can transfer from their native language: partial repeats and understanding checks. An example of an understanding check is included below. This type of repair strategy is similar a repair strategy that is often used in native discourse, which is referred to as the (*Huh?*) strategy. In the following example, in response to the learner's request for repetition, the entire trouble source is repeated by the interviewer:

Example 4

- Tutor: Was hat Ihnen dieses Semester im Deutschkurs ni:cht gefallen
What did you no:t like this semester in your German course
(1.5)
- Student: No:ch einmal?
O:nce more?
- Tutor: Mhm tch! Was hat Ihnen dieses Semester im Deutschkurs ni:cht gefallen.
Uh huh tz! What did you not like this semester in your German course.

CHAPTER 7. LEARNING STRATEGIES

Advanced learners of German are found to use one additional type of repair strategy: request for definition, translation or explanation (Liebscher, 2003). In their study they also found that students and teachers use very different repair types due to their role perception within the classroom. When interacting with a teacher, students show a preference for more specific repair initiation strategies. The following example illustrates this:

- Example 5**
- Tutor: f-fatima hat banken (.) gesagt
f-Fatima said (.) banks
 - Student: I DON'T UNDERSTAND WHAT gegen
obj
 - WHAT WE'RE TALK- gegenstand
object
 - Tutor: gegenstand ist ein objekt
'Gegenstand' is an object

The study by Cho and Larke (2010) analyzed conversation data of an ESL classroom in a suburban elementary school in Texas with five students (for which 24 classes were recorded). Similar to the adults of the previous study, the children's most frequently used repair strategies were understanding checks (117 occurrences) and partial repeats (144). Another interesting repair strategy found in this study is a request for repetition, which yields a repetition of the trouble source turn as response:

- Example 6**
- Tutor: ok. today I sneeze wobbly, yesterday I, what verb form
 - Student 1: uhm? one more time
 - Tutor: today I sneeze wobbly
 - Student 2: <sneezed>
 - Tutor: yeah sneezed.
 - Student 1: aha

In line 2 of the example, student 1 started his turn with two types of repair initiation: Unspecified repair (*uhm?*) was followed by a request for repetition (*one more time*). The tutor responded with the repetition of the trouble source in her turn in line 3.

A similar research study has been carried out to investigate the use of strategies to repair communication breakdowns by young French-English bilingual children of 2 and 3 years old (Comeau, Genesee, & Mendelson, 2007). The children interacted with an experimenter who only spoke a single language during the entire conversation. When they would start using the other language the experimenter would make up to five requests for clarification. All children in this study were capable of switching languages after language breakdowns and some even initiated self-repairs as soon as they noticed they were using the wrong language (see Examples 7 and 7 with a 2 and a 3 year old):

- Example 7** - TAL (2;9):
TAL : La la soeur. The the sister. [pause] The sister.
- Example 8** - JIA (3;2):
JIA: Un a. [pause] A bird.

7.4 Conclusion

Learning strategies are perhaps the most important component of the student agent's architecture. By making use of the meta-level architecture of language processing, they construct an agent's construction inventory and tweak the configurations of the grammar engine. Similar to flexibility strategies, a learning strategy is always built around a problem that occurs during routine language processing, thereby activating the meta-level. Because learning is inherent to the presence of the meta-level architecture, an agent can exclusively learn when a problem is encountered and learning proceeds very fast, even through a single problem repair. This problem-based approach brings meta-level learning strategies closer to the ideas that were explored in the constructivist learning theories of Piaget and Vygotsky. The next chapter demonstrates the learning strategies for the language system of Spanish verb conjugation, as the student and the language agent are interacting with each other in language games.

Chapter 8

A student agent for Spanish verbs

The only difference between a student agent and a language agent is the use of the meta-level architecture. While a language agent makes use of *flexibility strategies* to correctly diagnose and repair processing problems, a student agent employs a set of predefined *learning strategies* to *expand and adapt its construction inventory and grammar engine configurations* while engaging in interactions with the language agent. This chapter demonstrates how these learning strategies become operational for the domain of Spanish verb conjugation, as a student agent and a Spanish language agent interact in language games. Figure 8.1 illustrates this language game with a diagram of the processes a student agent runs through on the top and the language agent's processes at the bottom. All processes that are flagged carry potential for learning. While in parsing and production the student agent can modify its construction inventory, learning strategies that are active in the consolidation process at the end of every language game can potentially make changes to all three components of the student agent's architecture.

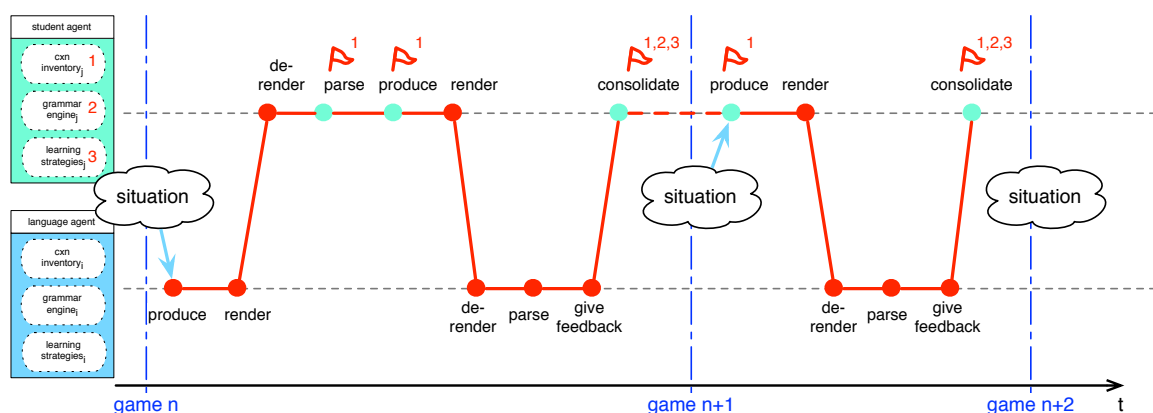


Figure 8.1 – The student agent can model the acquisition of the target language system by running through a number of processes in the tutoring game and learning when processing problems are encountered.

Unlike the Spanish grammar bootstrapping technique, which was introduced above as a shortcut to quickly create a large-size target grammar, learning strategies for Spanish verbs do not depart from an infinitive but instead encounter conjugated verb forms. They are used by a student agent to acquire the target language systems through situated interactions (also called *language games*). A learning strategy, as Chapter 7 has already explained, is a plan to tackle - in this case - a language acquisition problem that occurs during online processing. This problem-driven plan always consists of a set of *diagnostics and repairs* to identify and solve the problem in question. A strategy contains probability scores on the edges that lead from a problem to its possible repairs that indicate which repairs are most probable to occur after the problem has been diagnosed.

Although a student agent's construction inventory starts off as an empty list ready to be filled up with constructions, learning strategies are assumed to be aware of the general linguistic patterns of the Spanish language. Such patterns are readily observable by every adult learner with a background in another Indo-European language and a minor exposure to the Spanish language. They include knowledge about the composition of a verb form (i. e. a stem preceding a suffix) and the features that are expressed by a suffix (person, number, tense, aspect and mood). Also the rules of stem and affix formation are assumed shared knowledge by language learners and are therefore "hard-coded" in the learning strategies.

Learning strategies take the general linguistic patterns of a language into account.

Box 8.1 – Learning strategies assumptions

The student agent learns through interacting with a language agent that has full competence of the target language. Learning strategies are active at the moment of parsing verb forms produced by the language agent or when producing verb forms that express a certain conceptualized meaning. This section first explains the language game that lays out the scene for verb conjugation learning in Section 8.1. Section 8.2 contains a detailed description of the learning strategies that the student agent employs to acquire different aspects of Spanish verb conjugation. Special attention is given to the learning of verb classes. Finally, the results of the use of learning strategies to construct a grammar for Spanish verbs are included and discussed in Section 8.2.4.

8.1 The language game

In this language game, the language agent and the student agent interact and play so-called *reference games*, in which the speaker selects a topic from the situation that he then describes. The hearer's task consists in finding back the topic after interpreting



Figure 8.2 – An example of a learning situation in *Rosetta Stone* shows how the learner can select one of two contrasting answers (indicative/subjunctive mood) for this sentence. The correct answer here is to use the subjunctive form of *tocar* 'to play' "toque" ('I don't think she plays the guitar well').

the speaker's utterance. Either the student or the language agent acts as the speaker. The situation that they share contains two events, which only differ in their temporal, aspectual or modal specifications (see Figure 8.2). The speaker then selects one of the two events as the topic and produces an utterance (a verb form) that describes this topic as appropriate as possible. The hearer's task is then to parse the utterance and retrieve the topic from the situation, without guessing.

A set of six contrasting tense/aspect/mood specifications is used to automatically generate situations:

1. present vs. present perfect (e.g. *canto* vs. *he cantado*);
2. present indicative vs. present subjunctive (e.g. *canto* vs. *cante*);
3. present perfect vs. past perfect (e.g. *he cantado* vs. *canté*);
4. past perfect vs. past imperfect (e.g. *canté* vs. *cantaba*);
5. present vs. future (e.g. *canto* vs. *cantaré*); and
6. past imperfect vs. conditional (e.g. *cantaba* vs. *cantaría*).

A language game succeeds if the hearer could retrieve the topic that was selected by the speaker. It fails otherwise. Because the student agent is randomly assigned the role of speaker or hearer he can encounter problems in production and in parsing. When the student agent acts as the speaker of the game, it frequently occurs that he produces

an utterance that is not completely grammatical. The language agent then detects the irregularity through his flexibility strategies and reproduces the utterance in a corrected form. Now, the student agent is informed that his original utterance was incomplete and he can use the corrected utterance to infer what went wrong. No explicit feedback is provided. This inference mechanism is part of the learning strategies that form the topic of the next section.

The first game in Figure 8.1 illustrates this with the tutor as the speaker. The student agent's first process here is parsing the tutor's description. In the beginning, parsing will always fail because the student agent does not yet have the appropriate constructions. When this happens, learning strategies are consulted that aid in expanding the construction inventory (1) or modifying certain constructions so that they can apply to parse the utterance at stake. The conceptualization that is readily available to the student agent can be used here to (re)construct the necessary form-meaning mappings. Learning implies that you test what you just learned, not only in re-parsing the tutor's utterance but also in trying to reproduce what the tutor agent has just said. Therefore, in the second process on the time line, the student agent runs a production process that yields in his own utterance to describe the situation. Again, this process might need the assistance of learning strategies that control the assembly of the construction inventory.

The tutor agent determines the success of the current game by checking whether the student agent's utterance is the same as his own. Success or failure is the only form of feedback that the student agent receives in this game. The student agent now proceeds to the last process of the game where he consolidates the recently acquired knowledge so that possible misconceptions can be restored. Consolidation relies on learning strategies that can modify all three components of the student agent's architecture: (1) construction scores can be reinforced or inhibited or unknown feature values might be filled in; (2) configurations in the grammar engine might be changed when some inefficiencies were detected or (3) the scores of learning strategies themselves can be adapted here. In consolidation, the student agent has access to all constructions that were learned and used in the game. After this updating has taken place, the student agent is ready to start the next game.

Game 2 in Figure 8.1 shows an example of when the tutor acts as the speaker in the game. Because it is assumed here that the target grammar expresses the full conceptualization of the situation that is shared, the student agent will start a production process that tries to cover all elements (expressed as predicates) in the conceptualization. Obviously, in real language users – apart from not sharing the same conceptualization – do mostly only express parts of what they conceptualize in a given situation. On the way to producing an utterance, the student agent might encounter a range of problems that are diagnosed by the learning strategies he has been supplied with. Frequent problems that occur at the initial stages of a tutoring session concern the inexpressibility of certain predicates in the conceptualization.

The produced utterance is subsequently parsed by the tutor agent (not shown in the figure). The tutor's parsing can be seen as an evaluation of the student's utterance. Successful parsing does not always lead to a positive evaluation. Even if the student utterance is grammatical and can therefore be parsed it might not be an adequate description of the situation. This adequacy is verified through a re-production of the conceptualized meaning by the tutor agent and a successive comparison between the two utterances. When they are the same, the game succeeds, otherwise it fails. The reproduced tutor utterance is always sent to the student agent, who parses it to match the intended meaning. Here, again, learning strategies can operate that modify the construction inventory when problems are diagnosed during parsing. Finally, consolidation solidifies learned constructions and can modify the grammar engine or learning strategy scores.

8.2 Learning problems under the looking glass

The problems that a student agent encounters when learning how to conjugate Spanish verbs vary in terms of complexity levels. Although the semantic complexity of Spanish verb conjugation in the use of tense, aspect and mood is far from trivial, the learning strategies that are implemented in this first cycle focus on the learning of formal constraints. We Learning basic endings of a regular verb is an easier job than figuring out when to change a verb stem such as “volv-” (< *volver*, 'to return') into “vuelv-” or when to assimilate a verb stem with its ending and when not (e.g. *actúo* 'I act' vs. *evacuo* 'I evacuate'). An additional complexity is added for figuring out which suffix belongs to which verb class: e.g. “-aba” (1st) vs. “-ía” (2nd or 3rd). Moreover, the number of verb classes is not disclosed to the learner. This section goes through these three levels of complexity in the acquisition of Spanish verb conjugation step by step and homes in on some specific problems that learning strategies are working on.

The student agent's learning strategies are specialized on the acquisition of the formal features of the Spanish verb system.

Box 8.2 – Learning formal constraints.

8.2.1 Learning the basic mappings

To construct basic knowledge of the Spanish verb grammar, a student agent relies on a couple of learning strategies that help him to handle new verb stems or suffixes that he has not encountered before. The main idea is that he uses the situation in which he

hears the unknown forms to figure out their meaning and usage pattern. Because every situation deals with one event that is pictured in two different constellations (e.g. present perfect versus present tense), the meaning of an unknown stem is easily retrieved. Two slightly more complex learning strategies, the learning of irregular verbs and the use of syntactic suffixes are discussed in this section.

Once again, the psychological reality of these learning strategies is not at stake here, I am only after a system that can learn Spanish verb conjugation in a way so that it can communicate error-free. By building such a system, I want to gain a better understanding of the difficulties that are involved in learning this particular language system and the learning strategies that could tackle these.

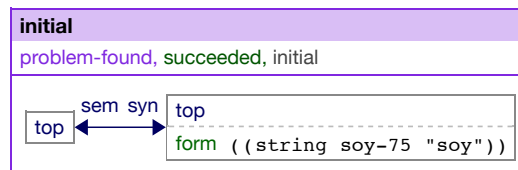
Irregular verb problem

Irregular verb forms are memorized as single chunks, that cannot be subdivided into a stem and one or more suffixes (see Section 4.2 on the Spanish construction inventory). As a part of his learning strategy knowledge, the student agent can discern verb stems from endings. Therefore, when no such segmentation can be made, the irregular verb form diagnostic triggers automatically and returns an **irregular-verb** problem. This problem is then tackled by a repair that specializes on it, in this case the **learn-irregular-verb** repair.

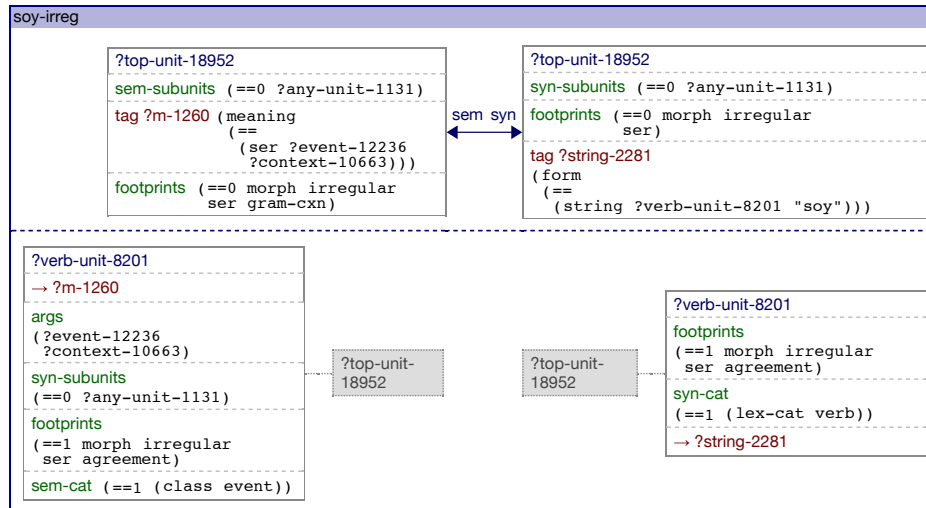
The first repair that tries to solve the problem of an irregular verb form takes the verb form that is unknown, for instance *soy* 'I am' and relates it to the events present in the learning situation: e.g. the to-be event in the 1 sg present indicative vs. 1sg future indicative. When the repair can figure out which of the two event meanings belongs to *soy*, meaning predicates are created and a new irregular lexical construction is added that links those to the unknown verb form *soy*. This inference can occur when the agent already has a construction for to be in the 1sg future indicative (*seré*). Or else, when a similar irregular construction is present in the construction inventory: e.g. *voy* 'I go' (also 1sg present indicative).

Else, when the student agent does not have any indications as to how to link the unknown string to one of the events in the situation, the repair adds a generic irregular construction that links *soy* to the event predicate only, without any further specifications (Figure 8.3b). As a consequence, the student agent cannot disambiguate the topic event from the situation (guessing is not permitted) and the game fails. The **missing-meaning** diagnostic triggers now. Yet, as a result of this failure, the speaker points to the event that he described with *soy* and another repair **add-meaning-after-pointing** takes the meaning related to the topic event and inserts it into the recently created irregular construction (see Figure 8.3c) so that it is complete.

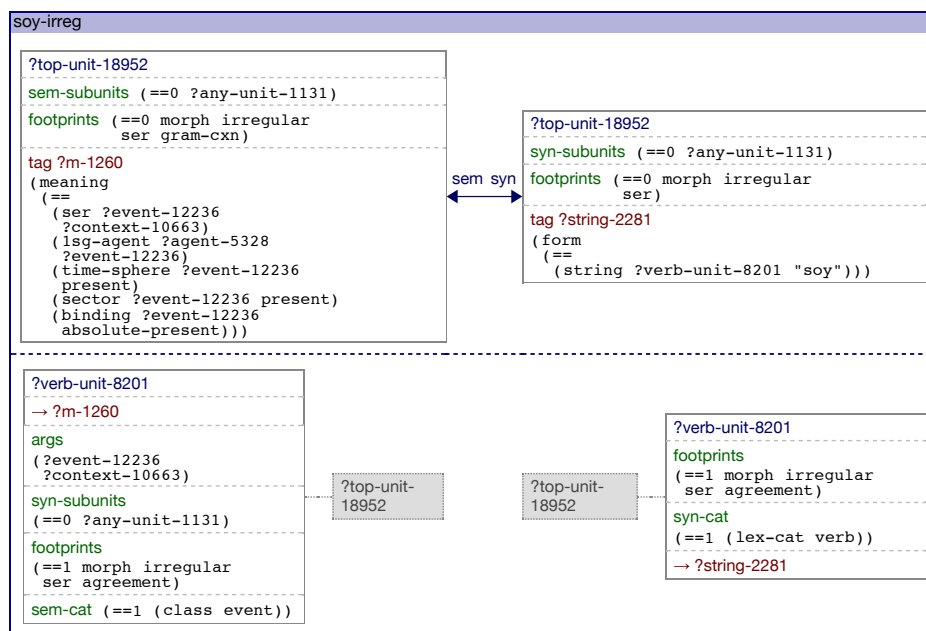
8.2. LEARNING PROBLEMS UNDER THE LOOKING GLASS



(a) Problem found



(b) Soy construction with general meaning



(c) Soy construction with complete irregular meaning

Figure 8.3 – The learning strategy for irregular verbs proceeds in two steps: once an unsegmentable string problem is detected (a), a general lexical construction linking the string with the infinitive meaning is added to the construction inventory (b) but after the speaker has pointed to the correct event, this construction is replace with a complete irregular lexical construction (c).

One could argue that this approach is similar to the dual route to learning verb conjugations, similar to the original proposal by Rumelhart and McClelland (1986) who argued that irregulars are memorised as complete chunks without internal rules and regulars can productively be created following certain grammar rules. Yet, Spanish inflectional morphology (similar to the past tense in English) exhibits so-called *quasi-regularity* (or semi-regularity), which is captured by the verb type footprints and the phonological constructions for stem changes. Quasi-regularity can be caught by the decision tree that was presented in Chapter 6. Only suppletive items such as the primary verbs in the Onoma decision tree fail to use any of the constructions that produce regular tense conjugations (McClelland & Patterson, 2002).

Unknown verb ending problem

When a verb form could be segmented by the student agent, it consists of a stem and an ending (the ending is not further segmented, unlike the language agent's verb form segmentation): e.g. *habl -an* 'they talk'. A number of scenarios are possible when the student agent parses this segmentations:

- A lexical construction for the stem is lacking in the student agent's construction inventory: e.g. *habl* cannot be parsed.
- A morphological construction for the ending is lacking: e.g. *-an* cannot be parsed.
- Both the stem and the ending are unknown to the student agent.
- Constructions exist for both stem and ending but their feature structures are incompatible: e.g. *-an* is only known for verbs of the 1st verb class, while *habl* carries a 2nd verb class feature.

I will focus here on the second scenario: the unknown verb ending problem. In this case, the student agent has a lexical construction for *hablar* 'to speak' but lacks a morphological construction for the *-an* ending. Three main learning strategies are involved in the acquisition of verbal endings, focused around three different learning problems:

1. Unknown ending: The ending string cannot be parsed by the grammar engine due to the lack of a morphological construction for it (diagnostic). The repair adds a new construction that links the unknown string *-an* to a generic meaning variable *?meaning*, an intermediate solution until the ending's meaning can be filled in.
2. Missing meaning: The new morphological construction is used but does not yield a definite meaning that disambiguates the topic event in the situation (diagnostic). The repair operates after the speaker's pointing action, when the missing

meaning can be inserted into the morphological construction. Notice here that, different from the typical morphological constructions, the ending construction is a **sem-syn** construction: linking grammatical meaning to a string (Figure 8.5a).

3. Variable inequality: The new **an-suffix** construction can now be used to parse the *hablan* verb form completely but the parsed meaning contains a variable inequality in the event variables:

```
((3pl-agent ?agent-4331 ?event-10196)
 (time-sphere ?event-10196 present)
 (sector ?event-10196 present)
 (binding ?event-10196 absolute-present)
 (hablar ?event-10192 ?context-8961))
```

To link the **hablar** event variable to the remaining event variables, a phrasal construction needs to be introduced that represents this link explicitly. This phrasal construction adds the grammatical meaning that was before added by the morphological construction directly to the verb unit and links it through the **args** feature (Figure 8.4). The meaning feature can then also be removed from the morphological construction, which thereby turns into a **syn-syn** construction that links grammatical features about tense-aspect-mood (represented by the tam-matrix) and person-number (person/number-matrix) to the ending *-an* (Figure 8.5b).

8.2.2 Stem changes

Stem changes are very frequent in Spanish and they occur in a variety of forms, which have been explained in detail in Chapter 4. Also the decision tree that was used to classify Spanish infinitives in the previous section heavily relied on indications for stem changes that the infinitive could induce. But how does a real learner construct these subtle differences in use between “cuento” ‘I tell’ and “contamos” ‘we tell’?

Because the student agent hears both stem variants, he will have saved two different lexical constructions for the meaning of ‘to tell’: the **cuent-cxn** and the **cont-cxn**. At first, there is no explicit relation established in the grammar between these two verb constructions. It is only when the student selects the wrong stem for a specific ending that a problem is first revealed. The learning strategy that is activated when the original stem is corrected by the language agent adds a new irregular construction for the corrected verb form. Because irregular constructions have higher construction scores, they are always tried first in production so when they can match on the complete conceptualized meaning that the student agent wants to express, they will immediately trigger and express the full verb form in one go.

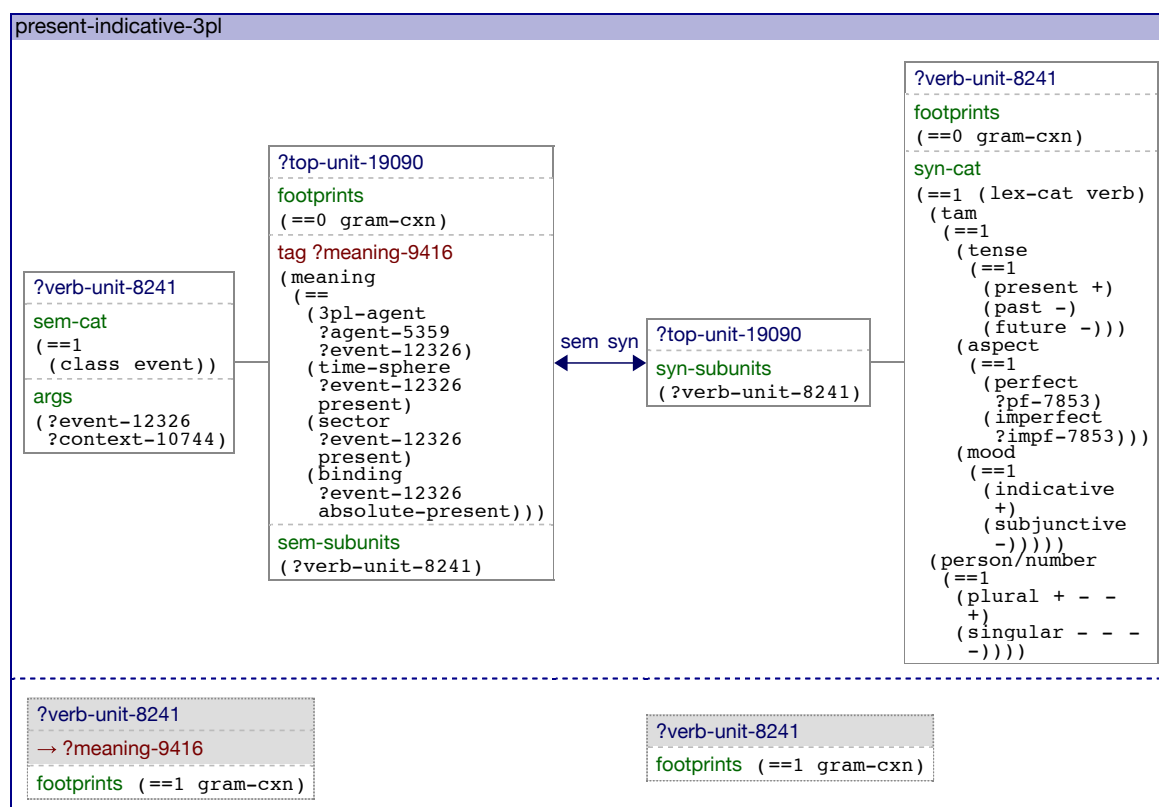
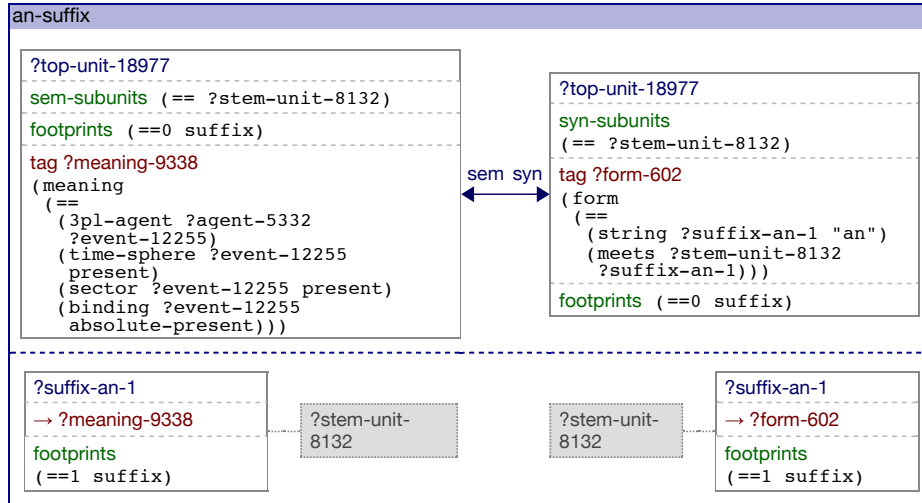


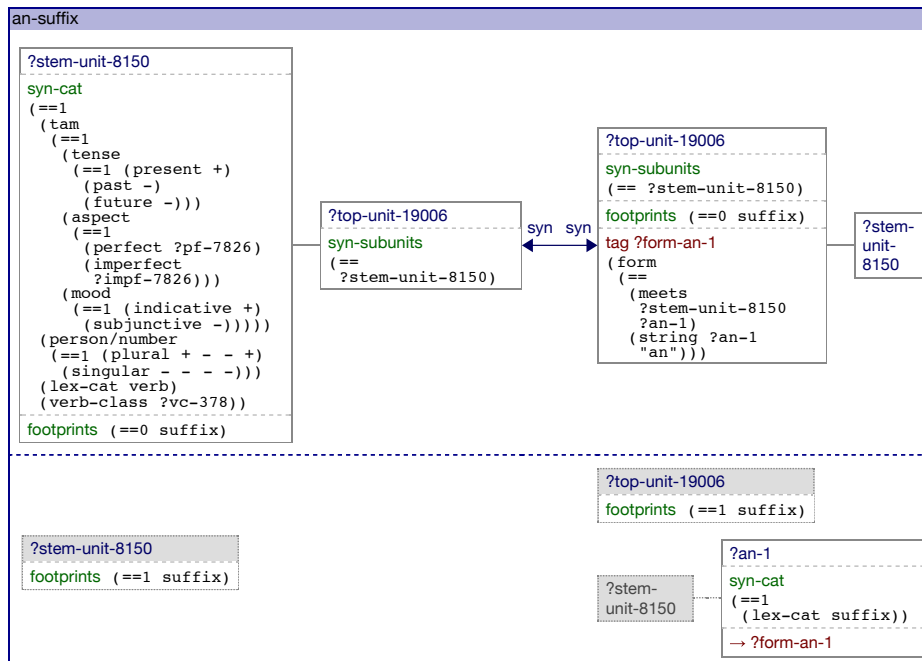
Figure 8.4 – A grammatical construction is added to link the meaning expressed by the verb ending to the actual verb unit.

Generalizations over the memorized stem-suffix combinations are possible once enough information is gathered and the semantic and syntactic information present in these constructions can be analyzed and compared. This has not yet been done in the current implementation of the Spanish verb learning strategies but it is an extension of the current stem-change learning strategy that does not only make use of local situated information but extends over a whole range of games and constructions that have been learned at different occasions.

In a future cycle of development, semi-regular verb conjugations will be learned by making use of the decision tree employed by the language agent to bootstrap a verb conjugation paradigm. With small adaptations to enable the tree to classify conjugated verb forms with a certain degree of certainty ("mislearnings" can be adapted later), the student agent would be able to learn the conjugation patterns of semi-regular verbs as well as the distinction between regulars and semi-regulars. Finally, the decision tree could also be used to diagnose missing knowledge in the student agent when it is used as a student model. The chapter on future outlook (Chapter 9) discusses this possibility in further details.



(a) -an construction with a meaning predicate



(b) -an construction as a syn-syn construction with a tam matrix

Figure 8.5 – A morphological construction for a verb form ending goes through two main stages: (a) one stage where the grammatical meaning is directly linked to the suffix string, (b) and another one where this meaning is replaced with syntactic information that the meaning expresses.

```

for w = stem
for s = suffix
if no class for w and s
do if s in history  $w_j \in Cw_j$ 
    do  $s \rightarrow Cw_j$ 
    else do invent class:  $s \rightarrow Cs_{n+1}$ 
                         $w \rightarrow Cw_{n+1}$ 
else if  $w \in Cw_i$  and no class for s
    do coerce:  $s \rightarrow Cw_i$ 
else if  $s \in Cs_i$  and no class for w
    do coerce:  $w \rightarrow Cs_i$ 
else if overgeneralisation error:  $Cw_i \Leftrightarrow Cs_i$ 
    do  $s \rightarrow Cs_i^-$ 
else if  $Cw_i$  and  $Cs_i^-$ 
    do invent class:  $s \rightarrow Cs_{n+1}$ 
                         $w \rightarrow Cw_{n+1}$ 

```

Figure 8.6 – Pseudocode for the learning strategy that tackles the verb class learning problem.

8.2.3 Verb classes

Similar to its ancestor Latin and its sister languages French and Italian, the Spanish language divides its verbs into three main classes depending on the endings of their infinitives: *-ar*, *-er* and *-ir*. Depending on the class a verb belongs to, it will combine with different suffixes to form a conjugated verb form. For instance, *cortar* 'to cut' is joined with *-aba* to form the past imperfect, whereas *beber* 'to drink' goes together with *-ía* to express the same tense/aspect information. Yet, verbs belong to the *-er* (2nd) or *-ir* (3rd) verb class share most of their endings and only differ in two forms in the present tense (*bebemos* vs. *vivimos*; *bebeís* vs. *vivís*) and the forms they use for the future and conditional tenses. This problem of ending sharing is also known as the *overlap* problem in machine learning, where a large number features are shared between two or more classes so that training data points are no longer linearly separable (Bishop, 2006, p. 331).

However, the student agent will never hear an infinitive in a language game and does not know which ending to use with which stem. He has to learn the possible endings for

a verb stem by extracting information from verb forms that are parsed, either in games when the language agent is the speaker or when the utterance he produced has been corrected by the language agent. When verb constructions and suffix constructions are first learned, no information is added on the verb class they adhere to. Similar to learning stem changes, the first moment when the student agent realizes he needs a constraint on the way of combining verb stems and endings is when he misproduced an utterance he thought to be accurate. A typical example of this scenario is when the student agent would produce **beban* 'they drink' instead of *beben*. He thus reuses the *-an* ending that he learned earlier in parsing and uses it in combination with the *beb* stem to express a 3rd singular present indicative tense of the verb *beber*, 'to drink'. Yet, when he already had the *-en* ending in his construction inventory but had a choice of selecting *-en* or *-an* the verb class learning strategy creates a first verb class and associates the verb stem *beb* as well as the correct ending *-en* with it.

The pseudocode in Figure 8.6 contains the five cases that the student agent can encounter when the verb class problem triggers. Because the student agent does not know how many verb classes there are in the target language, he can learn arbitrarily many classes: $\{C_0, \dots, C_m\}$. Every word, or verb stem w can belong to exactly one class: $w \in C_i$. A suffix s can belong to one or more classes but can at the same time also explicitly be a negative member of other classes: $s \in C_i, \dots, C_m$ and $s \in C_j, \dots, C_k^-$. There are three main processes at work when the student agent is learning verb classes:

1. **Category application/invention:** A new verb class is invented when both the word w and its suffix s do not yet have a verb class feature yet and the verb form that the student agent originally produced was corrected by the language agent. Before inventing a new class, the learning strategy verifies whether other suffixes that have been used correctly with the same word in the past already carry a verb class feature. If this is the case, w and s also adopt this verb class. Otherwise, a new class is invented and w and s receive a new verb category feature on its syntactic construction pole: e.g. (verb-cat (==1 (1 +))). As soon as there is more than one verb class in the running, verb constructions receive also negative verb category for the verb classes they do not belong to: (verb-cat (==1 (1 +) (2 -))). This expansion ensures that suffix constructions with a (2 +) feature are not used together with a stem of the first verb class.
2. **Coercion:** A verb class feature can be coerced into the feature structure of a word w or a suffix s when one of these carries a positive verb class feature. New verbs are thus added to a particular class as soon as they are combined with one of the class' endings.
3. **Overgeneralization:** Because the second and the third verb classes share a lot of their endings, the student agent is likely to have a single class for verbs that actually belong to different classes. I refer to this effect as an overgeneralization error. The

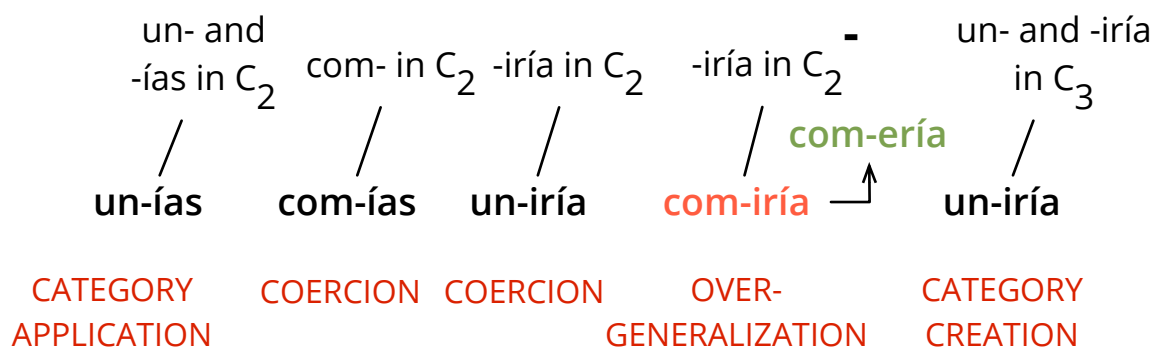


Figure 8.7 – An overgeneralization error occurs when the student agent wrongly believes that a verb stem and a suffix belong to one particular verb class (e.g. C_2) and is corrected by the language agent: e.g. **comiría* → *comería*. When this happens the original suffix construction receives a negative verb class feature value. Such a negative value is in a future application the trigger for creating a new category (final step).

real challenge lies thus in splitting one existing verb class into two classes. Figure 8.7 explains this process in further details, why it occurs and how it is handled.

Sometimes four verb categories are needed to get rid of misclassifications at an early stage in the learning process. The example below shows how a two class-system splits into a four class system within the course of 12 interactions. Because class 2 contains a mix of stems of all three Spanish verb classes: *tratar*, *violar* (1); *unir*, *vivir* (3) and *comer* (2). The four class system is then refined until there are no longer "mixed" classes but we end up with a divide of the *-ar* verbs into two verb classes: 2 and 4. Yet however, the local learning strategy does not pay attention to merging two verb classes into one when all their suffixes are shared.

```

500: (1 . vend)
531: (1 . vend) (2 . viol)
544: (1 . vend) (2 trat viol)
560: (1 . vend) (2 trat viol un viv com)
561: (1 . vend) (2 trat viol viv com) (3 . un)
572: (1 . vend) (2 viol viv com) (3 . un) (4 . trat)
823: (1 com vend) (2 viol viv) (3 . un) (4 . trat)
1141: (1 com vend) (2 . viol) (3 viv un) (4 . trat)
4999: (1 com vend) (2 . viol) (3 viv un) (4 . trat)

```

These unnecessary verb class divisions can be attributed to the way in which a new verb class is instantiated. When there is a feature value mismatch that reveals a positive sign for the verb stem and a negative sign for the verb ending (e.g. (2 +) vs. (2 -)), a new verb class is created unless the ending has been used with another verb stem that has a different verb class (e.g. 3) or the verb stem has an alternative form which has

Table 8.1 – Verb class distribution in the Jehle corpus grammar

<i>-ar</i> (1)	<i>-er</i> (2)	<i>-ir</i> (3)
0.65	0.17	0.17

already been classified. In this case, the verb and the suffix constructions both receive a verb category feature for the third verb class.

8.2.4 Results

The main measure in these student-agent/language-agent language games is communicative success, irrespective of who is the speaker in the language game. It is only when the student agent can parse a verb form to its complete meaning that he can also retrieve from the situation that a game succeeds. Or, in production, when the verb form produced by the student agent can be parsed by the language agent without resorting to his flexibility strategies and signaling a processing error. When the stage is reached where all games succeed, the student agent has obtained the highest level possible for the given game set-up and we speak of *convergence*. Therefore, inverting the communicative success curve would show us the error rate that the student agent is obtaining.

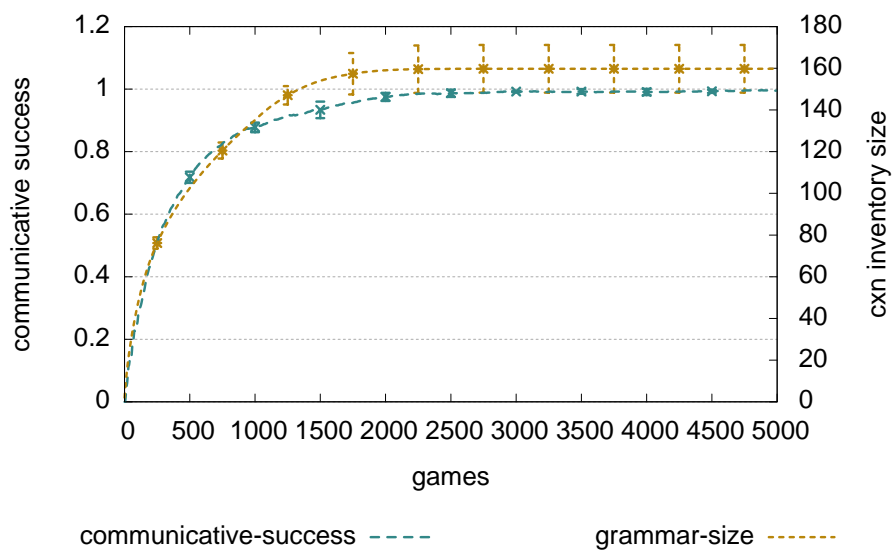
Yet, success in verb class learning is directly related to the input distribution: uniform, with all three classes equally frequent or nonuniform with one class more than double as frequent than the two others. When we use the complete Spanish test grammar (with the 600 most common infinitives) 65% of all verbs belong to the *-ar* class and the remaining *-er* and *-ir* classes are equally frequent and account each for 17% of the verb inventory (Table 8.1).

The discrimination process that underlies the acquisition of a particular verb class categorization directly depends on the verb input distribution (class, type).

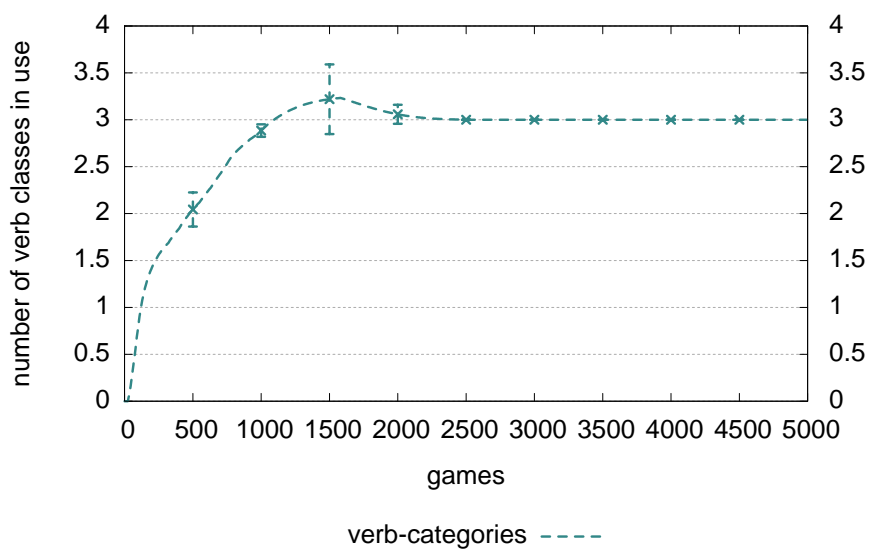
Box 8.3 – Learning verb classes

How effective are the learning strategies?

To first test the effectivity of the student agent's learning strategies I started with *learning the full conjugational paradigms of six regular verbs*, with an equal distribution over the three verb classes: (1) *tratar*, *violar*; (2) *vender*, *comer* and (3) *vivir*, *unir*. Figure 8.8 shows the results for an average of 4 x 5000 language games. Convergence



(a) Communicative success



(b) Number of verb classes

Figure 8.8 – To learn six regular verb conjugations (equally divided over the three Spanish verb classes) in discriminative contexts a student agent needs roughly 160 constructions in its inventory. Communicative success rises to 100% as soon as the construction inventory size and the number of verb categories have stabilized (Input = 6 regular verbs. Error bars= Average of 4 game series, standard deviation.).

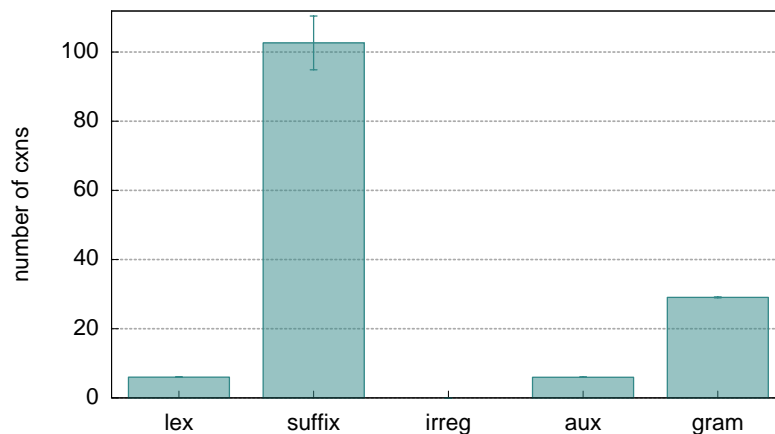


Figure 8.9 – The construction inventory for six regular verbs contains 6 lexical constructions, no irregular constructions, six auxiliary constructions, 30 grammatical constructions and a varying number of suffix constructions that is averaged around 102.

reaches 100% as soon as the maximum number of constructions is reached (a), which coincides with the moment in which the verb class number is settled at three (b). As this figure shows, the number of verb classes can mount to four for a brief period of time, which is sometimes required to split one verb class into two new classes: e.g. (vender vivir comer unir) => (vender vivir comer) (unir) => (vender vivir) (comer) (unir) => (comer vender) (vivir unir).

The final grammar size when learning the conjugations of six regular verbs fluctuates between 150 and 170 constructions. Figure 8.9 shows that this difference lies in the number of suffix constructions that are invented. Because verb class 2 and verb class 3 share most of their endings, some suffix constructions do not have a positive verb category feature but only specify that they can not be used after a 1st verb class stem: (verb-cat (==1 (1 -))). Instead of having two different suffix constructions, the student agent can thus successfully use only one construction.

Now, when more than six verbs are used and also *irregular and semi-regular infinitives* are allowed, the resulting picture becomes less easy to analyse clearly. To avoid a complete explosion of the verb classes, I randomly select 25 infinitives from the Jehle grammar and investigate how the student agent is able to acquire their conjugation. The 25 infinitives are selected every time a new interaction series is started and differ thus between multiple runs. Figure 8.10a shows an example of how verb categories can emerge and develop within the first 1000 games of a new interaction series. The stems associated with each class are included. Eight classes are instantiated: verb cat 1 and 3 represent the first verb class in Spanish (-ar), verb cat 4 contains verbs ending on -ir and the remaining verb classes all contain single verb instances. These single classes were created when the verb stem was used with an ending that was not yet categorized (or had a negative

feature value). Currently, no means to merge two or more verb categories into a single one exist. Figure 8.10b displays the number of suffix constructions per invented verb category: the largest categories for the *-ar* verbs have the biggest inventory but the remaining categories are following rapidly.

Communicative success still increases rapidly when learning the conjugations of 25 verbs but full convergence is reached relatively late. Figure 8.11 shows a long phase of continuously reshaping the construction inventory with new suffix constructions to cover all verb classes and new irregular constructions that express stem changes. Still, although new constructions are occasionally still being learned, most games are successful already as the communicative success curve demonstrates (small error bars). As expected, the construction types for which the number of constructions varies are suffix constructions and irregular constructions (Figure 8.12a). Their number depends on the discrimination process that categorizes the input verb forms as well as on the distribution of 25 verbs that was randomly selected for the game series (regular, irregulars, verb classes). The total number of verb classes fluctuates around 9 (Figure 8.12b) and they are introduced within the first 500 games of a new series. In the remaining games of the series, new verbs can be added to existing verb classes and suffix constructions are categorized into one or multiple classes.

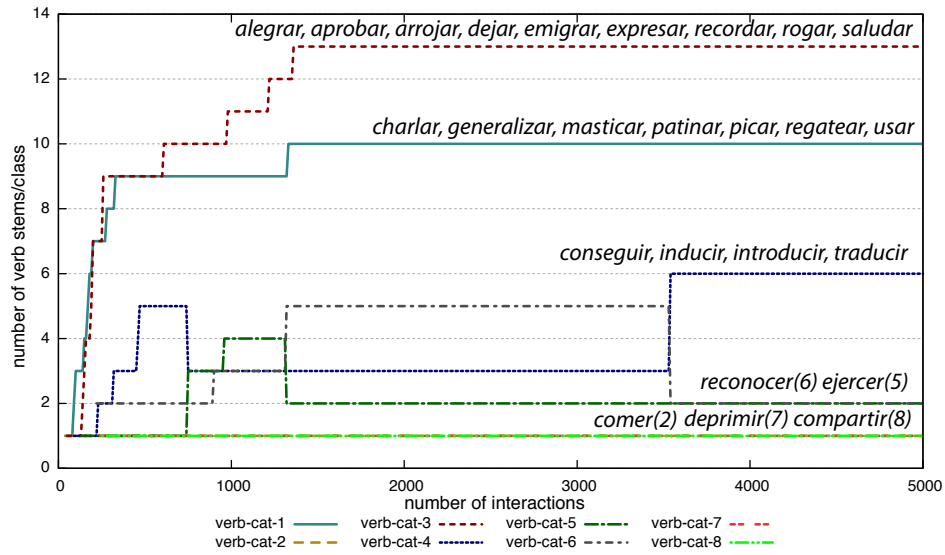
Reshaping of the construction inventory continues even after a student agent's communicative success scores have been maximised. Variants of suffix constructions (different verb class, same form and function) or irregular verb forms are still being acquired throughout the series of 20 000 games (25 input verbs).

Box 8.4 – Verb class reshaping continues after 100% communicative success.

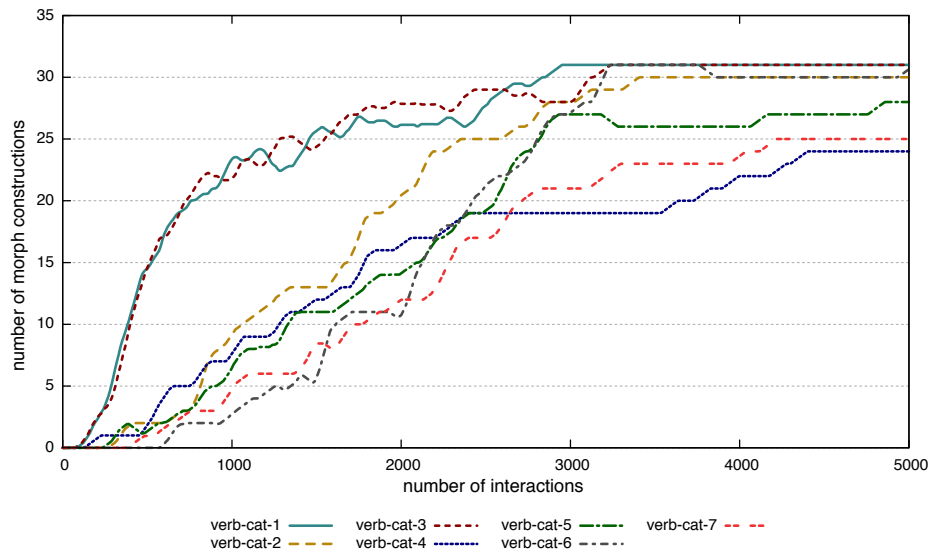
The three main learning stages that a student agent runs through can be summarized as follows:

1. **Item-based learning:** Verb forms are segmented but suffixes are attributed a full semantic specification as they express part of the conceptualization made by the speaker. Lexical constructions for lemmas can be learned after the language agent (expert) has pointed to the topic event that was expressed in the utterance.
2. **Abstract learning:** Grammatical constructions are learned in this stage, because they link lexical meaning of a lemma with the grammatical meaning expressed by a suffix. Verb classes are being learned now, as soon as the student agent makes mistakes in production by combining a particular stem with a suffix that belongs to a different verb class.
3. **Verb class reshaping:** All basic constructions are into place now and the student

8.2. LEARNING PROBLEMS UNDER THE LOOKING GLASS



(a) Number of verb classes



(b) Suffix constructions/verb class

Figure 8.10 – The development of verb classes happens quickly once the student agent starts to produce utterances with verb class inconsistencies. Productive classes such as the *-ar* class (verb-cat-1 and verb-cat-3) are first to reach a stable suffix inventory. (Input = 25 verbs. Single game series run.)

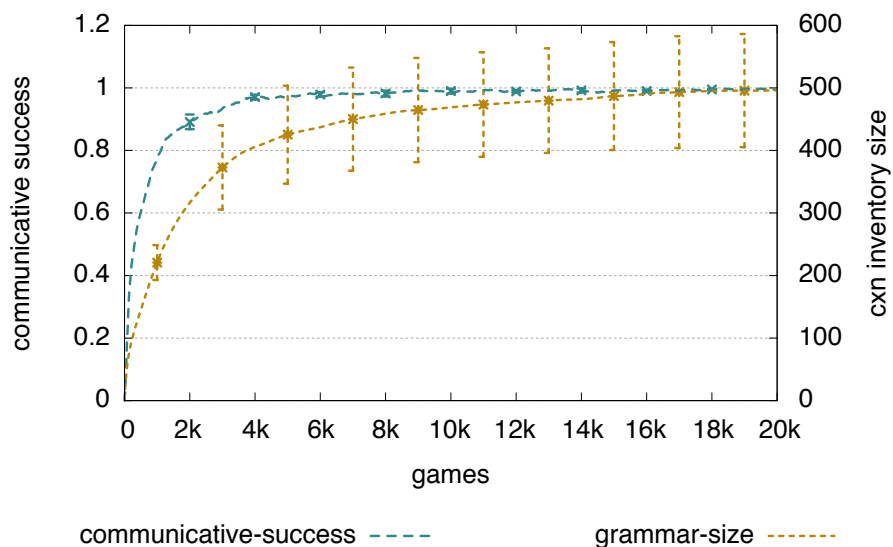


Figure 8.11 – Although the construction inventory size keeps slowly increasing until it finds a steady state, communicative success reaches (Input = 25 verbs. Error bars= Average of 10 game series, standard deviation.)

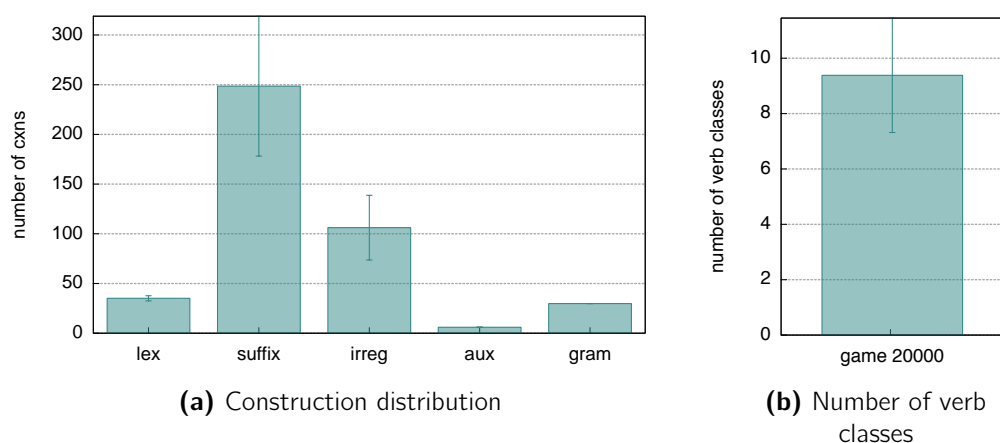


Figure 8.12 – The distribution of construction types in the resulting construction inventories reveals that 50% of the grammar consists of suffix constructions (a). Irregular verb constructions capture verb forms that cannot be expressed by combining a regular stem and a suffix. The final number of verb classes mounts to 9 classes with a standard deviation of 1.2.

agent can communicate almost flawlessly. Lexical constructions that have not yet been categorized into one of the existing verb classes receive a verb-cat feature and suffix constructions are shuffled around different verb classes to accommodate for new verb-cat feature combinations.

What is the effect of the early student production?

Apart from the input set of verbs that is used to create new learning situations, another experimental parameter that can be tweaked in this experimental set-up is the moment when a student agent is allowed to start taking up the role of the speaker, rather than only listening. Because there is a considerable amount of learning that happens when the agent is allowed to produce utterances (with learning strategies that especially target production), my hypothesis is that it is beneficial for the student agent to start to speak as soon as he can. To investigate the effect of the initial moment of speaking, I have tested three measures: communicative success, the construction inventory size and the number of verb classes used by the student agent. Apart from the default set-up where the student agent starts to speak at game 0, two additional starting moments have been tested: after 500 games and after 2000 games.

Because errors that a student agent makes during speaking help him to learn more efficiently, the moment of the first production should come as early in the learning process as possible.

Box 8.5 – Hypothesis: Early speaking improves learning.

Although one would expect that the delay of this moment would influence the rate of *communicative success*, Figure 8.13a shows that this hypothesis can only partially be validated. The final communicative success scores (game 5000) for the three scenarios is always 100%. Yet, after the introduction of speaking, the score drops sharply (speaking 500/2000) but it starts to rise again at a slower rate after the agent has recovered the largest share of the "mislearned" constructions. When the student agent is purely listening and absorbing information, the communicative success score reaches 100% considerably faster because competence in listening requires less learning effort than complementary speaking and listening proficiency.

Yet, interestingly enough, the *number of constructions* that the student agent knows differs between the different speaking onsets and does not seem to have an effect on the communicative success (Figure 8.13b). The average total number of constructions in the student agent's construction inventory is only slightly larger when the agent starts to speak at game 500 (± 15 constructions) but its number is grows more when speaking is delayed until game 2000 (± 30 constructions).

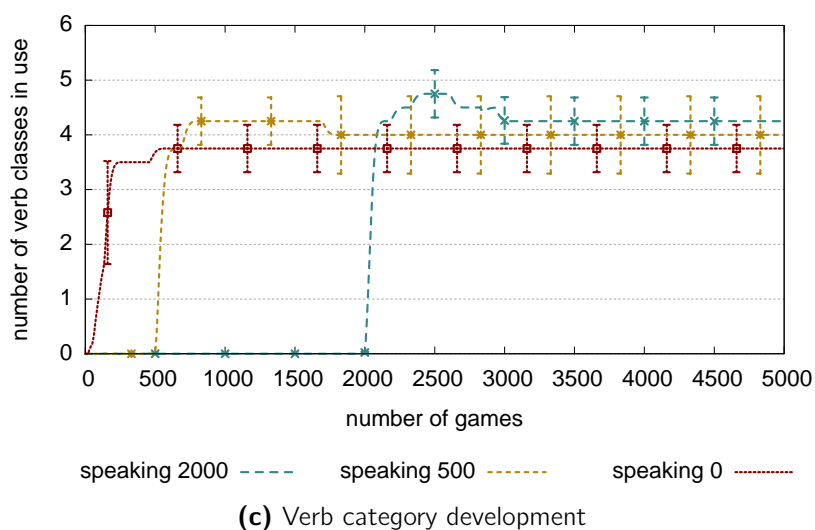
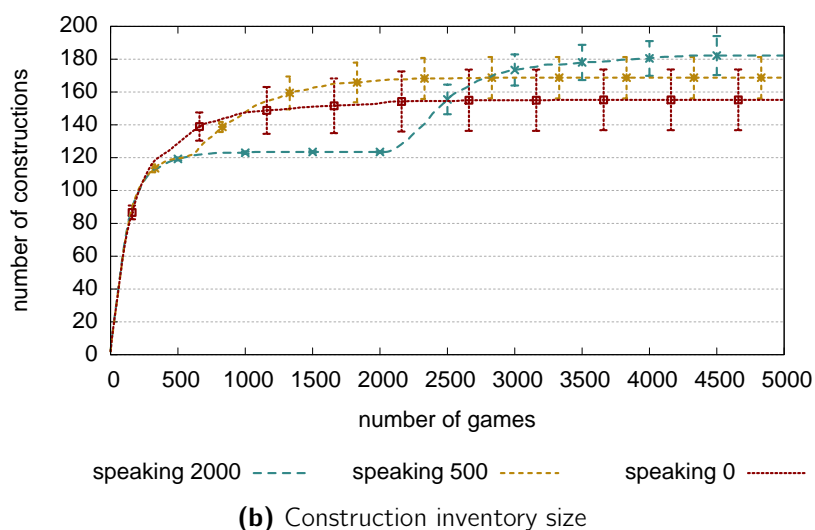
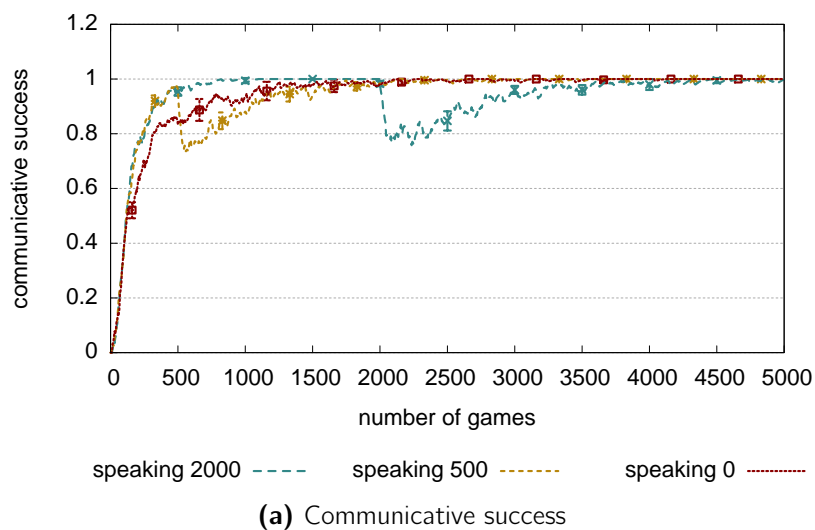


Figure 8.13 – The moment at which the student agent starts to speak only has an effect on the size of the construction inventory (b) but does not affect the communicative success rate. (Input = 6 regular verbs. Error bars= Average of 10 game series, standard deviation.)

Also the *number of verb classes* that is present in the construction inventory is slightly higher when speaking is delayed considerably in the learning process (Figure 8.13c). Although the final number of verb classes in use fluctuates around 4 for all three conditions (3.8, 4 and 4.2), the late speakers create a small overshoot of verb classes that need to be unlearned during the game series. This additional verb class that was introduced during learning but then never used again also explains the difference in grammar sizes depicted in Figure 8.13b.

The hypothesis that speaking early is beneficial for learning cannot be corroborated clearly in these experiments. To really get a good estimate of what is happening when the moment of speaking is delayed, more parameters should be tested and a more detailed analysis of the constructional learning is needed. Nevertheless, because no significant difference could be shown between the moments of speaking, the default parameter setting remains the scenario where the student agent and the language agent have an equal chance to take up the role of the speaker from game 0.

Construction competition

Language games with a population of agents that is larger than 2, as the current game has, automatically face the scenario where different solutions are invented in two independent games. The competition that results from multiple inventions for the same grammatical feature or concept is usually dampened using a lateral inhibition strategy. Wellens (2012) gives a good overview of strategies that have been used in the Naming Game since its first appearance. Yet, with only two agents in a Naming Game, this type of ambiguity (multiple forms for one meaning) does not occur. Also, the student agent does not invent new forms when he cannot express a particular meaning but gives up and waits for corrective input from the language agent. Previous experiments have looked at analogical learning methods that can be used to invent a new form when related forms already exist in the construction inventory of an agent.

Instead, learning only happens after correct input has been received, which implies that there is no ungrammatical invention possible on the part of the student agent. Also the acquisition of Spanish verb classes does never lead to real competition between constructions. When a new class is created, a stem construction's verb class feature is simply overwritten. Although a suffix construction is copied with a change in the verb class feature but the copy is no real competitor of the original construction. The original suffix construction is also still correctly used in combination with particular stems.

In sum, scores are not used here because the learning experiment is a Naming Game with two agents, of which one is an expert speaker. Only constructions such as irregulars receive a score that is higher than lexical constructions for stems so that they are tried first in processing, a property that is inherent to the learning strategy that invents them.

8.3 Conclusion

Compared to the bootstrapped grammar of the language agent that could quickly be created and covers a large number of verb paradigms almost without processing errors, the student agent's grammar is much more costly to construct and amplifying the number of verbal conjugations that can be learned increases the learning time considerably. Yet, as this chapter presented a first study of how we can get an insight into the difficulty of learning Spanish verb conjugation, the learning strategies that were used to achieve 100% communicative success can be improved in future cycles as to decrease the learning time. Also the final number of constructions in the student agent's construction inventory does not scale linearly with the number of input verbs. Due to the locality of the learning strategies, the more input verbs, the more verb classes will need to be learned. This issue should also be addressed in future work.

However, the student agent cannot yet be treated as a student model. To really use the student agent as a model of a student in a tutoring game there are two operations needed:

1. Potential gaps in a student agent's grammar should be made tractable so that they can be used by tutoring strategies. To do, this a student agent should also be able to invent new suffixes in production based on analogical reasoning over suffixes that express related meanings.
2. The student agent's grammar needs to be aligned to the real student's knowledge of the target grammar.

The next chapter discusses the future outlook of how the student agent can be used to incorporate these tasks.

Part III

Future outlook

Chapter 9

Future outlook

With a lot of time and effort put into the development of a *robust language agent* that can parse and produce any verb form of any verbal paradigm of the Spanish language, including ungrammatical forms, and a *student agent* that learns new constructions incrementally by interacting with a competent language agent, it is now time to explore how these two key ingredients can be combined in the design of a first prototype of a language tutoring system for Spanish verbs. Yet, before considering such a design, let us first review the achievements made with the implementation of a language and a student agent for Spanish verb conjugation:

1. A decision tree can be used to capture the underlying rules of Spanish verb conjugation patterns. It classifies infinitives into a particular pattern depending on formal features they exhibit in their verb stem or theme vowel.
2. Together with a small set of basic constructions to capture verb endings and grammatical constructions for tense, aspect, mood and agreement, a full Spanish grammar can be bootstrapped from this decision tree. The end nodes of the tree call particular construction templates that create constructions based on the categorisation that is made.
3. A student agent can induce constructions based on data and also bootstrap a grammar for the Spanish verb conjugation language system. It does not learn from infinitives that it classifies, but it receives instead fully conjugated verb forms in a discriminative context.

However, our student agent is not yet a student model in this first implementation. The results in Chapter 8 demonstrate the difficulties in learning Spanish verbs. They do not make claims about how human learning happens but show a first attempt to learn

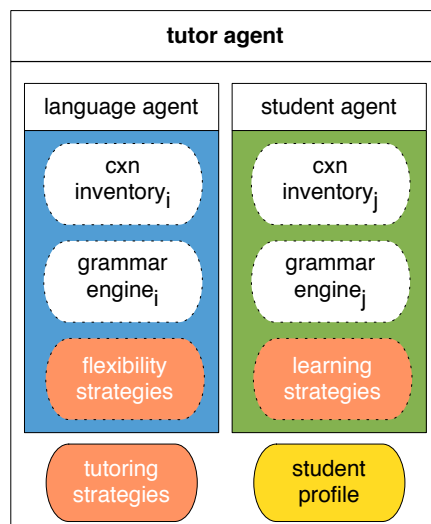


Figure 9.1 – A tutor agent has a language agent and a student agent at its disposal as well as two additional components: a student profile and a set of tutoring strategies.

the target language system based on data with the main goal to gain insight into *how an agent learns*. Now that we have such a learning agent, the creation of a student model comes within reach. A student model would be able to say which constructions or concepts are still lacking in the student agent's grammar, which could be used by tutoring strategies as diagnostics to select future language games that specifically address such a gap.

The tutor agent's three types of strategies for flexibility, learning and tutoring, all rely on the same meta-level architecture with diagnostics and repairs that solve problems.

Box 9.1 – The tutor agent's meta-level architecture

This chapter describes the *design* of a future tutoring system that evolves around the concept of a tutor agent (see Figure 9.1). Two essential elements of this design are the student model that the tutor agent employs and the tutoring strategies that he has at his disposal. Tutoring strategies are included in the design of a prototype for an agent-based tutoring system to steer the learning curve of the student and select new situations that are challenging enough (but not too demanding at the same time). To fully be informed about the current knowledge and the skills of the student that is being tutored, the design of a tutor agent includes one last component: a student profile. This student profile contains static information about the student such as a user profile and measures about his level and learning goals.

The outline of this chapter is fourfold. First, Section ?? provides some background on the larger pedagogical vision in which the tutor agent will be developed in future work. Second, the student model is explained as an extension of the existing student agent in Section 9.2. Third, Section 9.3 describes how tutoring strategies can be operationalized. Finally, Section 9.4 includes a first illustration of an interaction with the tutor agent for Spanish verbs.

9.1 Pedagogical vision

Apart from Intelligent Tutoring Systems, researchers in the field of Artificial Intelligence have also experimented with building the right learning environments for computer-based education. Therefore, when launching the tutor agent as a kind of personal language tutor that helps you tackle the acquisition of a particular language system, we should consider the ideal learning environment it will act in and how it will maintain the learner's motivation to play more language games. This section discusses types of learning environments in Section 9.1.1, arguing for a trend towards situated learning in open-ended learning environments. Section 9.1.2 focuses on learner motivation in pedagogy and introduces the autotelic principle in the theory of flow. Yet, what follows is not intended to be a complete account of the pedagogical principles that are involved in these topics but rather provides a sketch of what the tutor agent should one day become.

9.1.1 Learning environments

When we think of learning environments, most of us will be reminded of typical environments that are made for classical instruction such as a classroom in a school building or the piano chair in the conservatory where you spend many of the evenings during your youth. But for others a learning environment can be their grandmother's vegetable garden, a friend's chemistry laboratory or the supermarket in a foreign country. The best learning environments are probably those of which the learner is not aware, that stem from natural situations in which learning happens very quickly (Krashen & Terrell, 1983).

This section gives an overview of a range of learning environments that we encounter today and their consequences for the effectiveness of learning. It starts with the classical approach to education in Section 9.1.1, which has been dominant for hundreds of years in our Western culture. Section 9.1.1 explores the theory of *Situated Learning*, which was coined by Lave and Wenger in the early nineties to stress the importance of using the student's actual environment for learning. Finally, Section 9.1.1 reviews open-ended learning environments that help students externalizing their ideas (Papert, 1980)

and point to the role of tutors to speed up self-directed learning in such environments (Vygotsky, 1978).

The classical approach

For hundreds of years, teachers have been presenting knowledge in carefully organized packages to students that passively receive this information and work independently to learn from fixed tasks from century-old curricula. As Park Woolf put it, "[t]hese passive methods suggest that a student's task is to absorb explicit concepts and exhibit this understanding in largely factual and definition-based multiple-choice examinations" (Park Woolf, 2008, p. 14). Teachers typically ask 95% of all questions in the classroom, which require short answers or problem-solving activities (Graesser & Person, 1994; Hmelo-Silver, 2004).

Yet, these traditional teaching methods are not very effective (Waterman, Matlin, & P.A., 1993), as they usually only succeed with the top fourth of each class, which often includes the most motivated and talented students. As Bloom's two-sigma problem has shown (see Figure 2.1), student achievement can be improved by two standard deviations when students are tutored individually rather than instructed through classroom interactions (Bloom, 1984). But the student/teacher ratio is not the only factor that determines the learning success. Learning environments also play a considerably large role.

It was widely assumed that to implement effective teaching methods that actively engage their students technology would be needed. However, as the history of intelligent tutoring systems has shown, the early "teaching machines" were all based on so-called "programmed instruction", a teaching method where the student interacts with the computer according to a pre-programmed script which may allow for some limited, and predefined, variations. These programmed-instruction systems almost always result in similar ineffective learning outcomes as the traditional classroom teaching methods. Due to the inflexibility of these programs, for instance, if the student re-runs a particular lesson, he is liable to be presented with the same materials/dialogue as he encountered on previous occasions.

Situated learning

The theory of *Situated Learning* was developed by Lave and Wenger (1991) who argued that "learning is more than a transmission of abstract knowledge from one person to another but it is situated in certain forms of social coparticipation". Situated learning is the most natural form of learning that occurs spontaneously when children are learning concepts and linguistic expressions. Instead of teaching about vegetables in the classroom

with pictures and descriptions, children can be taken to the local market in town to see, touch and taste vegetables that are typical for their region or for the season. Or, as a new PhD student, it is often more effective to really prepare a scientific presentation and present it in front of a real audience than when you prepare one for the purpose of an internal PhD course. You will automatically receive feedback on how you did and learn how to improve things for the next time.

Linda Smith and colleagues have extensively demonstrated that children acquire their first words in a situated context, by holding the objects in their hands and inspecting them very closely (Yu & L. B. Smith, 2012; Yu, L. Smith, & Pereira, 2008). But also when learning a second language in a natural setting, learners also employ situated learning. We learn words when we hear them in a particular context, we associate typical expressions with whom has used them and in which situations they were used, and we'll take on the pronunciations used by members of the community in which we are learning. Learning a second language will proceed in a similar way to learning a first language, immersed in the target language, and a communicative level can be bootstrapped quickly, especially when corrective or explanatory feedback is offered by other speakers.

Mobile technology is a very powerful tool to create situated learning situations. A photo blogging project initiated by Wong, Chin, Tan, and Liu (2010) involved students using smart phones to take photos that could illustrate Chinese idioms that they were learning and to share their photos and comments with the class through a wiki (Godwin-Jones, 2011). The effect on learning of using the student's actual environment has been shown to outperform projects that were based in a lab (Stockwell, 2008) or a classroom (Liu, Peng, Wu, Lin, et al., 2009). Another study that tested mobile technology for learning English as a second language in Dutch primary schools revealed that "formal school learning can be augmented by learning in an informal context, away from school" (Sandberg et al., 2011, p. 1334). In this study, students were given a mobile application for learning about animals in a public zoo (see Figure 9.2). A group of students that could take the mobile phone home for a fortnight showed a superior performance on the posttests that measured individual change in mastery of a set of targeted English words. Hence, the motivation to use the mobile application in their spare time really had a positive outcome on their learning progress.

Open-ended learning environments

The idea of an open-ended learning environments (OLEs) contrasts sharply with programmed instruction as they favor the teacher in helping to expand the views of learners, as an indirect process that relies on interaction and influences from the context and the environment of the learner. Open-ended learning goes back to Piaget's constructivism (Ackermann, 2004), who claimed that knowledge grows according to complex laws of self-organization, as we are learning through our actions in the world, and it can be



Figure 9.2 – Snapshots taken from (Sandberg et al., 2011, p. 1339): "On the left a multiple choice game about the Snowy Owl. On the right a jigsaw puzzle about the Elephant."

retrieved today in some of the computer-assisted personal learning environments that allow users to ask questions, "experiment, interpret and learn from errors to revise their knowledge" (Park Woolf, 2008, p. 17).

OLEs can potentially play an important role in sustaining someone's motivation. When you experience more freedom you can be more creative and adapt the learning material to your specific goals. When learning Spanish, perhaps you only want to know how to communicate with your new Spanish neighbors, or else you want to be able to read the sports section of a Spanish newspaper. OLEs help students to learn at their own pace, motivated by their own personal goals .

The most famous open-ended learning environment is probably LEGO, with its blocks that allow children to construct their own *microworld* (the term microworld was first coined by Feurzeig, of Computing Activities, Bolt, and Newman (1969)). A tool such as Lego supports self-directed learning through an iterative process by which "learners invent for themselves the tools and mediations that best support the exploration of intriguing ideas" (Ackermann, 2004, p. 20).

Inspired by the pedagogy of Lego, Seymour Papert developed the Logo turtle microworld (Papert, 1987), that employed the Logo programming language (Papert, 1980), which was developed to encourage students to think rigorously about mathematics not by teaching facts but by building meaningful products. You could talk to a turtle (sometimes shaped as a triangle) that inhabited a screen by typing commands on the keyboard. Some examples of commands and actual moves are included in Figure 9.3. A so-called

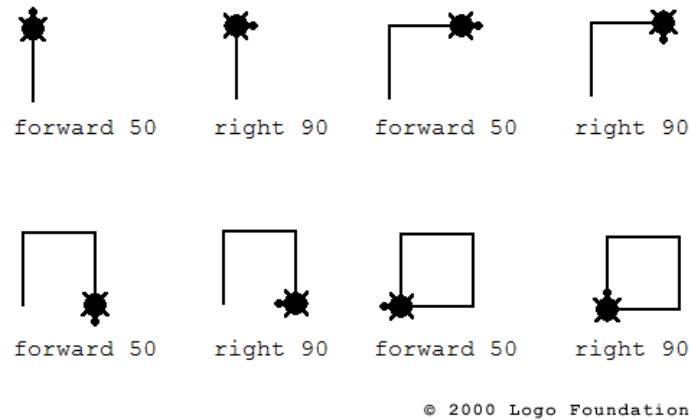


Figure 9.3 – By means of programmed instructions in the Logo programming language, students could move their turtle in its microworld and create drawings.

"microworld" represents a small slice of reality. "It is strictly limited, completely defined by the turtle and the ways it can be made to move and draw" (Papert, 1987, p. 80).

In open-ended learning environments, the learning process is generally more effective and faster when it is guided by a tutor with greater expertise. Similar to Piaget and Papert, Vygotsky also saw the child's intellectual development as a constructive process, but he put a greater emphasis on the role of culture as a teaching machine. Vygotsky coined the notion of *zone of proximal development* (ZPD) (Vygotsky, 1978). The ZPD establishes "a potential *area of expansion* that each individual has at their disposal to overcome their limits, provided the social environment in which the learning takes place "pitches in"" (Ackermann, 2004, p. 22). It tells us thus how far we can push the limits of what we know or can do, when helped by others.

9.1.2 Motivation

The motivation of many young people to participate in schooling activities, particularly in European urban and multi-cultural environments has become highly problematic, leading to stressful situations in classrooms and a high rate of drop-outs. School interactions that require high intellectual engagement but they are experienced by students as allowing a very low level of influence and therefore a low degree of intrinsic motivation (Della Fave & Bassi, 2003). This lack of student motivation in turn has led to a crisis in the motivation of teachers, who are forced to give standardised curricula that are too challenging for part of the student population, leading to student anxiety, and too easy for another part of the population leading to student boredom (Burke, 1996).

Maintaining and stimulating the learner's motivation is one of the most essential features of a tutor. Various types of motivation play a role when we want to reach a personal

CHAPTER 9. FUTURE OUTLOOK

learning goal such as playing a particular piece of music on the piano or running a half marathon: self motivation, peer motivation and social motivation (d'Inverno, 2012).

1. Self motivation can be driven by your desire to achieve something, by your curiosity to explore something new or by any other personal incentives.
2. Peer motivation relies on respect and trust you receive from others while you are learning. We are very often actively seeking for confirmation of our learning progress.
3. Social motivation originates from a personal wish to interact with others through our newly acquired skill.

Personal learning goals are typically driven by the learner's *intrinsic motivations*, when he creates his own learning incentive. Yet, instead of fostering intrinsic motivation, traditional motivational theories are based on the behaviorist idea of reward and punishment, and are thus promoting an *extrinsic motivation* in the learner. This theory is reflected in the *reinforcement learning*, where the learning process is accelerated by immediate feedback, acting as a reinforcer. Typically, the mildest reinforcer possible needs to be used, because, according to Lieberman (2000, p. 275) "if reinforcement is seen as a means of control for the benefit of the controller, then it is less likely to be effective". Reinforcement, also in its negative variant (punishment), is used in a number of CALL applications that give feedback by using praise words such as *excellent* or *brilliant*, or even *Wrong!*, *Incorrect!*, *False!* or simply *No!*.

On the other hand, cognitive approaches to learning have suggested that the most effective learning occurs when the learner is in a state of *flow*, a concept that was first introduced in the self-motivating theory first developed by Csikszentmihalyi (1978). Learners can reach optimal experiences when they find themselves in this stage and they are afterwards confronted with an intense feeling of happiness and self-realization. The search for the state of flow is an intrinsic motivation that can be induced, but that is fragile and can at the same time be destroyed. Flow is certainly something that can be learned, although it is also more than often suppressed.

This section further discusses instances of extrinsic and intrinsic motivation (including flow) and relates them to their use in CALL applications.

Extrinsic motivation

Extrinsic motivation comes from *outside of the individual*. Common extrinsic motivations are rewards such as grades or prizes that are obtained for showing the desired behavior. Punishment is usually threatened to follow any bad behavior. Reward and punishment

leads to the conditioning of a learner's behaviour, which will be solely relying on the presence of an external stimulus. In a competition, participants are encouraged to win and beat others, rather than to simply enjoy the intrinsic rewards of the activity itself. Not wanting to win can sometimes even be considered as disloyal behavior in sports. Our sports "heros" are nowadays rewarded with tremendous sums of money for their performances, which - not without surprise - does not always happen without any deceitful practices. Extrinsic incentives have sometimes lost all proportions and have surpassed the mere desire to hear a cheering crowd or to win a nice trophy.

There are a number of serious issues that can result from putting too much stress on extrinsic learning incentives, that do not originate from the personal interest of the learner, such as (p.c. Johan Loeckx):

- A reduction in the interest that a learner shows for the task in question;
- Learning happens much slower due to stress and because the learning goal is to obtain a high grade, rather than acquiring the content of the task or subject, making the learning process a meaningless activity (Dornbusch, Elworth, & Ritter, 1988; Gottfried, Fleming, & Gottfried, 1994);
- Learners often become uncertain and develop a lack of self-confidence, because they are expecting a third person (to whom they look up to) to confirm their learning progress;
- Creativity and the development of a critical viewpoint are ignored and considered as irrelevant for achieving the learning goal.

Yet, *extrinsic motivation is highly fostered in current CALL applications*. They generally inform you almost constantly about your scores (Rosetta Stone, Games for Language) or they let you lose hearts as soon as you make a mistake (Duolingo). Reward and punishment are omnipresent in these systems. They do not even try to find out what the user's real learning incentive is. Perhaps he skipped some parts and paid less attention to his answers because he was for instance not interested in learning how to communicate at the post office.

Intrinsic motivation

As Ryan and Deci (2000) state: "Students who are overly controlled not only lose initiative but also learn less well, especially when learning is complex or requires conceptual, creative processing". Intrinsic motivation, as opposed to extrinsic motivation, is defined as the doing of an activity because it is inherently interesting or enjoyable rather than for some separable consequence. Learners that are moved by an intrinsic motivation act

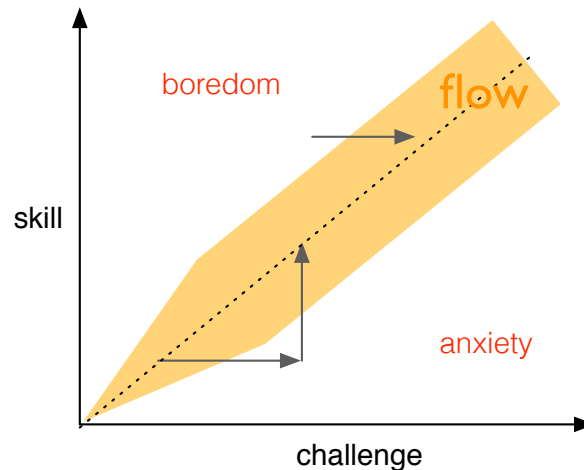


Figure 9.4 – The flow region is defined by a balance between challenges and skill. When the challenge is too low for your current skill level, you experience a feeling of boredom. By contrast, when your skill level is too low with respect to the challenge level, you experience a feeling of anxiety. This graph is a replication of the original flow chart.

for the fun or challenge that is entailed in the learning process rather than because of "external prods, pressures, or rewards" (id.). Such intrinsic motivation can be promoted by granting learners more freedom and abandoning systematic grading. In this way, the learner will be encouraged to explore domains of knowledge by himself, "instead of being supplied with pre-established answers" (Steels, 2004, p. 139). Examples where this approach has proven success - at least for pre-school or the first year of primary school - include specialized schools such as the Freinet schools (Sivell, 2004) and the Reggio Emilia experiment in Italy (Rinaldi, 2003).

Nevertheless, the question that imposes itself here is how we can prevent that learners find themselves in a completely unstructured and free learning environment, which can result in boredom and a general lack of motivation. The theory of self-motivated learning can help to formulate a tentative answer to this question. This theory was originally developed by Csikszentmihalyi (1978) who studied the activities of painters, rock climbers, surgeons and other people who are deeply involved in some highly complex activities without receiving a direct reward for it. In his theory, these activities are called "autotelic", meaning that the motivational driving force ("telos") lies inside the individual himself ("auto").

The strong form of enjoyment that is created by autotelic activities is been characterized as "flow" by Csikszentmihalyi. He intended a restricted use of this term, being a state that often occurs as a side effect of autotelic activities (Steels, 2004, p. 139):

People concentrate their attention on a limited stimulus field, forget personal problems, lose their sense of time and of themselves, feel competent and in control, and have a sense of harmony and union with their surroundings. (...)

a person enjoys what he or she is doing and ceases to worry about whether the activity will be productive and whether it will be rewarded. (Csikszentmihalyi, 1978, p. 182)

Yet, autotelic activities should be distinguished from directly pleasurable activities like going down a roller coaster. Csikszentmihalyi has argued that in order to make an activity autotelic there should be "a good balance between high challenge, generated through the activity and perceived as meaningful to the individual, and high skill required to cope with this challenge" (Steels, 2004, p. 140):

Common to all these forms of autotelic involvement is a matching of personal skills against a range of physical or symbolic opportunities for action that represent meaningful challenges to the individual. (Csikszentmihalyi, 1978, p. 181)

When engaged in autotelic activities, one gets sometimes leave the zone of flow and drift off into two directions: boredom or anxiety (see Figure 9.4). You get bored of an activity when your skill level is too advanced for the current challenges that are posed upon you. By contrast, a feeling of anxiety can occur when the challenge level is too elevated for your skillfulness. It is the role of a good teacher to keep a student in its region of flow to trigger a very powerful natural learning process that enables students to become self-motivated.

9.2 Student model

Any teacher who is involved with the learning process of his or her students relies on a conceptual student model of the skills and the knowledge that an individual student currently disposes of. Constructing such a model is a natural process, which occurs on a daily basis when parents interact with their kids, in a game situation when your opponent does not yet have the same skills as you or in a conversation with a non-native language user. In these situations you will try to gauge their level and progress so that you can automatically adapt your way of addressing them.

A student model also forms an indispensable component of any intelligent tutoring system that wants to be adaptive to its users. ITSs that lack student models are typically experienced as monotonic and impersonal by students. And yet, although many of the current commercial or academic tutoring applications are using a student model, they are still far removed from a complete simulation of the real student for the target learning domain. They can instead be compared to a school report that a real teacher would keep of a student, containing grades on different subjects but also information related to motivation, interests, cooperation in class, etc. To more closely approach a functional model

CHAPTER 9. FUTURE OUTLOOK

of the student's language learning process and constructional knowledge, the student model that the tutor agent uses makes the crucial distinction between a student profile and a student agent. The *student profile* is indeed like the "school report" used by many tutoring systems, a static record that keeps scores, preferences and other attributes. The *student agent*, instead, can be seen as an "incompetent" version of the language agent with its own construction inventory, grammar engine and learning strategies.

The student model is split up into two components: a static student profile for bookkeeping and an active student agent that simulates the real student.

Box 9.2 – Student model components

This section shows the benefits of this compositional student model architecture. It allows to split the main function of the student model, namely imitating the student, from less central bookkeeping concerns about the student's practice sessions and (needless) scores. I show that using the competent language agent as a mould for the student agent has many advantages, not only in terms of finding the optimal simulation of the student but it is also utterly cost efficient in terms of development. Being a usage-based student model, it expands its grammatical knowledge over time as more learning situations have been tackled. So although the language agent and the student agent share the same architecture, the student agent's construction inventory is not a simply subset of that of the language agent. Instead, it can be made up of different types of constructions or contain different values for certain key features, but also the mapping between hearing an utterance and interpreting it might not be completely optimized yet in the grammar engine.

The dimensions of a student model have been discussed in Chapter 2, Section 2.2.2: bandwidth, knowledge type and student-expert difference. Yet, linguistic knowledge is very different from knowledge about programming or arithmetics, domains which are typically handled by a student model. Although many ITSs treat linguistic competence as a number of rules to master, represented as a subset of the expert language (overlay models) or as a set of constraints (constraint-based models), I believe that language acquisition is a much more complex process and should be treated accordingly. There are a number of reasons to assume that a language learner's construction inventory is not a mere subset of the tutor's constructions:

1. The language learner will perhaps (or most likely) never reach the level of a fully competent language user with near-native capabilities because that is *not his main learning objective*. Many people learn languages to travel, to read books in their original version, to better understand a neighboring culture, etc.
2. When you learn a language *you aren't a blank slate* but you unavoidably bring the

constructions of other languages that you already speak into the learning process of a new language. Cognates are often very useful to be aware of because they can speed up the learning process considerably. Yet, false friends – pairs of words or phrases in two languages that look or sound similar but differ in meaning – might confuse the language learner and put him on the wrong track.

3. Even between speakers of the same language, construction inventories are far from perfectly aligned. Because *we all have different experiences and talk to different people*, our linguistic knowledge and skills diverge around a certain average performance, which can be compared to a prototypical language user of the language that is spoken in a community.

Although the language agent and the student agent share the same agent architecture, the student agent's construction inventory is not a simply subset of that of the language agent.

Box 9.3 – Shared agent architecture

Instead of highlighting the differences between student and expert (or tutor), the student model presented here ascribes to the *model-tracing approach* where the student model traces the real actions of the student to match his knowledge as close as possible but at the same time the model can be used to simulate the student's solutions and predict his future performances. The following sections outlines the student model's two-fold structure in Sections 9.2.1 (student agent) and 9.2.2 (student profile). Measures that are used by the tutor agent to estimate the level and the progress of the real student are introduced in Section 9.2.3.

9.2.1 The student agent

The student agent is the core part of the tutor agent's student model. Being a fully operational agent, the student agent is endowed with exactly the same components as a language agent: a construction inventory, a grammar engine and set of learning strategies. The case study in Chapter 8 has already shown the potential of using a student agent that can learn a target language from situated interactions. Yet, to become used as a student model, the student agent should simulate a real student's behaviour. It is private to every single student interacting with the system and no information is shared between students. Also, the student agent's internal structure and contents can only be inspected by the tutor agent but it is never shown to the real student. Because the student model does not assume psychological plausibility but intends to be

CHAPTER 9. FUTURE OUTLOOK

a functional model of the student, inspecting the student agent's constructions or the learning strategies would not be very informative to the student.

As Chapter 8 has shown, a student agent can interact with a competent language agent and acquire a particular language system without a single intervention of a student. All three components of a student agent can evolve over time, be it to different degrees. Before the first tutoring session has started, the student agent is initialized as follows:

1. **Construction inventory:** The list of constructions that is used to parse and produce utterances is initialized empty. This component is thus fully adaptive over time as more constructions can be added, deleted or modified. Modification can happen in the feature structures of a construction (adding/deleting a feature or changing a feature value) or in the score of the construction.
2. **Grammar engine:** The construction inventory's "motor" is initialized according to default configurations that are standard in FCG grammar implementations. Searching the list of constructions happens according to a simple list (no scores) and goal tests that check whether there no more constructions can apply are enabled. There is no construction-set or dependency-network organization available at the moment of initialization. Configurations can be modified or expanded (e.g. a construction sorting criterium can be added) through learning strategies. Yet, these modifications happen much less frequently than changes in the construction inventory.
3. **Learning strategies:** The student agent is initialized with a set of learning strategies that has been developed for the target language. In the current implementation, they do not evolve over time but there have been recent proposals to create evolving learning strategies (van Trijp, 2012).

Of course, a student model only makes sense when it closely matches the knowledge of the human student that is being tutored. Therefore, the student agent needs to be initialized at the student's level and aligned to it after every single interaction. In the first design plan of the student model, the initial state of the student agent is very basic and uninformed about the initial level of the student when he starts interacting with the tutoring system. Because there is currently no means to get an estimate of the student's level, the construction inventory is empty at the start of the games (see above). When the student is already an advanced learner of the target language system, his grammar should be bootstrapped quickly through his interactions with the tutor agent. Techniques to do this fall currently outside the scope of this dissertation.

Remember that the student agent should not just be a good model of the process of acquiring the target language system but it should more be a model of the real student's language acquisition. Therefore some constructions that it may have learned might have to be revised when the utterance of the real student is received by the tutor. The only

source of information (i.e. bandwidth) that the tutor has about the real student is his utterance, given a particular situation. There are two general actions that the tutor can undertake to use this information to align the student agent closer to the real student:

1. The real student's utterance can be parsed by the tutor agent and any errors that are found signal constructions that are not yet mastered.
2. The real student's utterance can be compared to the one produced by the student agent so that differences can be repaired and similarities reinforced.

The first action, error detection, is handled by the tutor's language agent, whose flexibility strategies contain tools to automatically spot errors, inconsistencies or novelties in a language user's utterances. We do this almost unconsciously – some people more than others – when we speak with children or non-native language users. Our ears are masters at detecting the most subtle distinctions in pronunciation, agreement patterns that do not fit the general rules of the language (e.g. a misplaced article use in Dutch such as *de meisje* instead of *het meisje*), etc. The tutor agent can thus similarly use his level of competence to judge the utterance produced by the real student and detect the type of error that was made.

The second action, alignment, is an issue that lies within the realm of the tutoring strategies that the tutor agent possesses. They will decide what to do with the error type that was detected in the erroneous input of the real student and how to handle differences between the student's and the student agent's utterances. Section 9.3 is dedicated to tutoring strategies. Such an update of the student model is traditionally seen in terms of *advancing* the student through the curriculum, step by step. Every time the student model was updated, the student's level was reevaluated to determine whether he could move on to the next learning stage (VanLehn, 1988). More recent proposals suggest that student knowledge can be updated through the use of stereotypical sequences of language acquisition. For instance, the sequences could mirror grammatical partitions that are typically found in textbooks in foreign language teaching (e.g. the CASTLE tutoring system; (Murphy & McTear, 1997)).

9.2.2 The student profile

Although usually referred to as the student model, a student profile merely keeps track of an individual student's performance. You can directly compare it to the student report that a primary school teacher keeps for every student in the class, containing a student's grades on different subjects over time, a score that denotes his motivation and cooperation in class, and perhaps also his rank amongst the other students. A student report is thus also inspectable by the student himself and by his parents. Yet, the student

CHAPTER 9. FUTURE OUTLOOK

profile that I propose here is not meant to be visible to the learner, but it could be if this is needed. Although ITS designers have argued for the use of open student models because learners like to compare their knowledge with their peers in the form of grades (Park Woolf, 2008, p. 54), I do not share this vision because it externalizes the learning motivation. Keeping the learner self-motivated is a much more important task for a tutoring system, and suggestions on how to sustain intrinsic learner motivation in the tutoring architecture proposed here are discussed in Section 9.3 on tutoring strategies.

Although the student profile – similar to the student agent – is hidden from the real student, the tutor agent is granted access over the student profile, which he partially helps to construct. The student profile can thus be defined as a database that records personal information about the student as well as the details of his interactions with the tutoring system. It contains four main elements (visualized by Figure 9.2.2):

1. personal user data,
2. a learning priority list,
3. logs with tutoring game data and
4. a range of measures that reflect the student's progression.

The student profile contains **descriptive** information about the student and his behaviour. Because there are no explicit scores, an explicit ranking between students is impossible.

Box 9.4 – Student profile

The student profile contains descriptive information about the student and his behaviour. There are thus no explicit scores for particular grammatical phenomena kept in the database. In this aspect, this student profile diverges much of the common student models in intelligent tutoring systems or computer assisted language learning research. Therefore, scores cannot be compared across multiple students and there is no explicit ranking possible. In a truly interactive system with multiple students, one could imagine to have students playing language games with more advanced students, selected based on their ranking. The following sections provide more details on the preliminary design plan of the four components of a student profile.

User data

A user profile is made for every new student that starts interacting with the tutor agent. It needs to be filled in by the student. This profile contains the student's user name

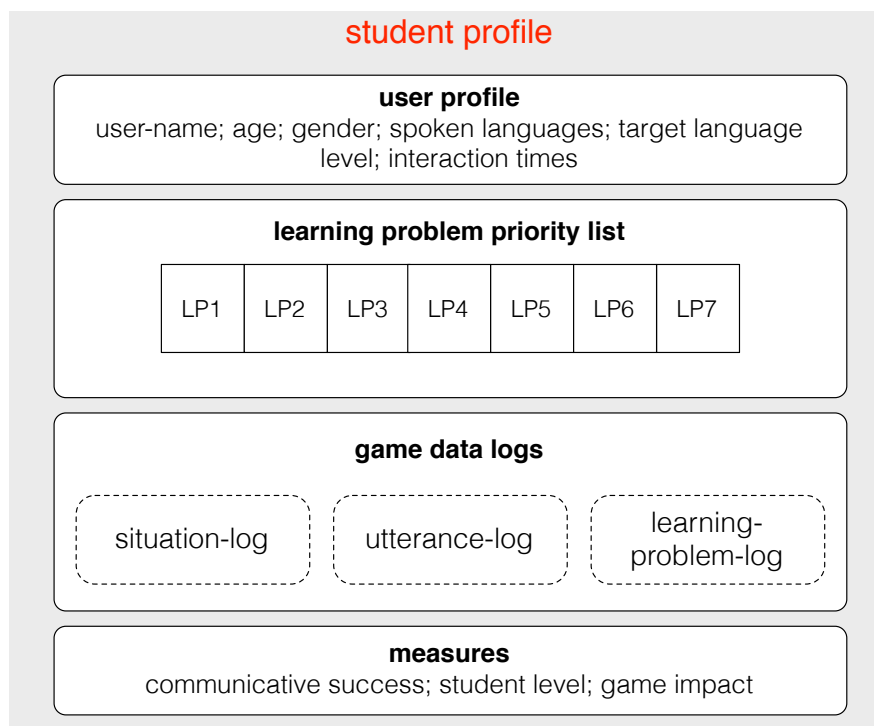


Figure 9.5 – The student profile consists of four main components that all record student-related information that is collected during a tutoring game. Tutoring strategies will use this information to make decisions when planning future tutoring games or providing feedback to the student.

(which serves as an id to identify users across multiple sessions), his gender, age, the languages that he already speaks (with an indication of the level of proficiency) and the competence he has in the target language. Although this information is not immediately used by the tutor agent, it can be interesting for analyzing and comparing the data of the learning process of different students. The first language parameter would be interesting to model transfer learning strategies, although this is not yet available in the current version of the tutor agent.

Learning problem queue

Every language system has a number of learning problems associated with it, which can be diagnosed by the learning strategies that a language agent or a student agent is endowed with. As we have seen before, the flexibility strategies of the competent language agent include advanced diagnostics that decide upon a particular error or slip that is encountered during parsing. These diagnostics signal learning problems that explicate the causes of irregularities that a student or student agent has produced. But also learning problems diagnosed by the student agent provide information about learning difficulties.

In this way, learning problems provide useful information about which parts of the lan-

guage system should be explored further in future tutoring games. They are therefore arranged in a *learning problem queue* (or LP-queue) that the tutor agent can consult when selecting a problem for a new learning situation. Problems that receive a high priority are inserted at the front-end of the queue. The LP-queue is updated by the tutor agent at the end of every game.

The learning problem queue contains problems diagnosed by the language agent or the student agent that should be revisited in future tutoring games. The queue is consulted by the tutor agent when a new learning problem is selected.

Box 9.5 – Learning problem queue

During one tutoring game, there are two types of learning problems that should be distinguished: the target learning problem(s) and the diagnosed learning problem(s). The target learning problem(s) are those that were popped from the LP-queue to create a new learning situation. The diagnosed learning problem(s) are problems that are diagnosed during the tutoring game, either by the student agent or by the language agent.

Game data logs

The tutor agent records everything that happens within one tutoring game in a variety of data logs. Logging happens automatically after every process that the tutor agent has successfully completed. Logs are consulted by the tutor agent's tutoring strategies where they assist the selection of a new learning situation (see Section 9.3.2). There are three main game data logs, which are connected through the game numbers:

1. **Situation log** (S-log): The conceptualized situation that the student agent and the language agent share is catalogued in the S-log. In case of a topic game, the topic is also recorded here. The S-log is consulted to avoid creating too similar situations within a particular window of tutoring games.
2. **Utterance log** (U-log): The U-log records the utterances of the student, student agent and language agent. Also the assignment of the discourse roles (speaker/-hearer) is kept here, where the student and the student agent are referred to as `student` and the language agent as `tutor`.
3. **Learning problem log** (LP-log): The LP-log registers the target learning problem as well as learning problems that were diagnosed during the tutoring game. It can be used to count the frequency of learning problems as well as their repetition in particular time spans (e.g. in the last 10 games, from game 100 - 150, etc.).

9.2.3 Measures

Measures keep track of the student's performance and progression over time. The measures that are kept in the student profile are recalculated at the end of every tutoring game. Instead of overwriting the previous value, the student profile keeps a list with all values of a particular measure, with the most recent one located first in the list. In this way, the tutor agent gets an estimate of how fast the student is making progress and when his performance has stabilized. These are the default measures that have been included in the design of the tutor agent:

1. **Communicative success:** This binary measure indicates whether the interaction between the student and the tutor agent was successful. Success occurs when the tutor agent can effectively parse the student's utterance or when the topic of the learning situation could be retrieved by the hearer (student or tutor agent). Communicative success is defined as follows:

$$CS = \begin{cases} \text{if game succeeds} & 1 \\ \text{otherwise} & 0 \end{cases} \quad (9.1)$$

2. **Feedback impact:** The impact of feedback that the tutor agent provides at the end of a game is calculated as the average of the confidence scores of the constructions that are affected by the feedback. Its values are situated in the interval ranging from 0 to 1. Feedback-affected constructions are those constructions that share the same features as the learning problem that is currently handled:

$$cxs_{affected} = \text{overlap}(cxs_{used}, LP_{features}) \quad (9.2)$$

The function *confidence* fetches a construction's confidence score and returns it as a number between 0 and 1. Feedback impact (FI) can then be defined as follows:

$$FI = \frac{\sum_{i \in cxs_{affected}} \text{confidence}(i)}{|cxs_{affected}|} \quad (9.3)$$

3. **Student level:** The student's mastery of the target language system is useful to get an estimate of how far he is from reaching the language agent's competence level. The measure therefore relies on the *overlap* function to calculate the similarities between the language agent's and the student agent's constructions. This similarity is then multiplied with the average confidence scores of the student

agent's constructions:

$$SL = \frac{\text{overlap}(cxns_{LA}, cxns_{SA})}{|cxns_{LA}|} * \frac{\sum_{i \in cxns_{SA}} \text{confidence}(i)}{|cxns_{SA}|} \quad (9.4)$$

Yet, there resides one danger in this measure. Because the student agent's construction inventory is empty at the start, the student's initial level will be set to 0. To match the level of students that start with some prior knowledge of the target language system, this measure should thus take a jumpstart, which can only happen when the student agent's construction inventory is constructed really in a fast pace.

Of course, the list of possible measures is long and depending on the objective of the tutoring system, other measures can be created and kept in the student profile.

9.3 Tutoring Strategies

Mirroring the flexibility strategies and the learning strategies discussed in Chapters 5 and 7, tutoring strategies use the same meta-level architecture of diagnostics and repairs to represent knowledge about the student. They perform two main functions:

1. guiding the interaction with the student through carefully selecting the game settings (game type, conceptualized meaning, topic and discourse roles) and
2. providing feedback that speeds up the learning process.

This section describes how the meta-level infrastructure could be turned into use for specifying tutoring strategies that support student model alignment and general game flow regulation. It shows when they are active during a tutoring game and how they interact with the student's actions. Using the same architecture brings a transparency into the construction of tutoring system, making it easier to manually compose and manage tutoring strategies. Accommodating the tutor agent with tutoring strategies has the potential of making the tutoring process personal and adapted to the individual student.

9.3.1 Reusing the meta-level architecture

As before, the diagnostics and repairs of the meta-level architecture operate on problems that are encountered during routine processing. In the case of tutoring strategies, these

problems are not related to misunderstandings or a lack of expressivity but instead they represent learning problems that the student is experiencing during a tutoring game or has faced in earlier games. As Section 9.2.2 has shown, learning problems are saved in a queue in the *student profile*, where they are ordered according to their priority scores. Yet, the diagnostics that signal a particular learner problem operate on information that is present in the *student agent*. They try to get an estimate of which constructions the student knows and how often he has successfully used them. Also errors that have been detected by the language agent's flexibility strategies module can be handled directly by tutoring strategies. Repairs intend to indirectly solve the learner problem by adjusting the tutoring approach that is taken or formulating constructive feedback.

Tutoring strategies tackle learning problems that a student has experienced by selecting appropriate learning situations and providing constructive feedback.

Box 9.6 – The function of tutoring strategies

Because they are strategies, tutoring strategies are used in the tutor agent's processes that involve decisions that have *a direct effect on the real student*. Within one language game, two processes have a direct link with the student:

- selecting the game's learning situation (at the beginning of a game) and
- providing constructive feedback (at the end of a game).

In between these two, a range of automatized processes take place, proceeding according to the same algorithms every time they are run. Figure 9.6 revisits the language game that was explored in Chapter 8 and visualizes all processes that make use of the meta-level with a filled triangle: learning strategies (student agent produce), flexibility strategies (language agent parse) and tutoring strategies (tutor agent select situation and give feedback). The language game that is illustrated in this figure is a description game where the student (uppermost dotted line) takes up the role of the speaker. He runs through three processes: produce answer (formulate an utterance that matches the topic he selected from the situation), receive feedback and adapt level (optional at the end of every game).

The tutor agent's processes are divided into three operational modules: a student agent (produce and consolidate), the language agent (parse the utterances of the student agent (utt-1) and the student (utt-2)) and the tutor agent. The tutor agent runs five additional processes, that are directly related to the interference of the real student in the tutoring game:

1. **Select situation:** the tutor agent selects the situation (and the topic in a reference

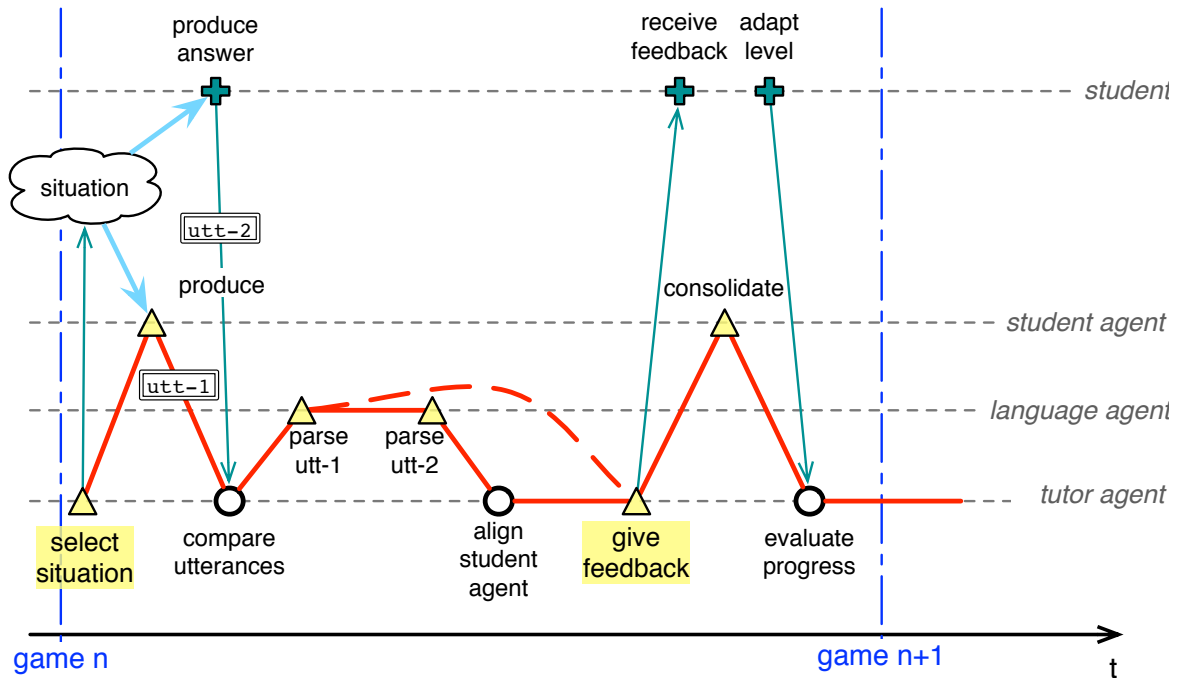


Figure 9.6 – The tutor agent runs the student agent, language agent and tutoring strategies during one tutoring game. Also the student profile is updated through the tutoring strategies module. Two utterances are received and depending on their similarities, either both are parsed by the language agent module (when different) or only the student agent’s utterance is parsed (when they were the same). Alignment only occurs when the student agent’s prediction was different from what the real student said.

game) for the present game based on the skill level of the student agent and problems that have been encountered earlier.

2. **Compare utterances:** the real student’s utterance is compared to the one produced by the student agent by a simple string matching process. When the strings are equal, the student agent managed to predict the real student’s behaviour and only one utterance is parsed. The tutor then skips two processes (parse utt-2 and align student agent), which is visualized by the dashed line. When the strings are different, both utterances are parsed by the language agent and the student agent is subsequently aligned to the real student by the tutoring strategies.
3. **Align student agent:** the tutor aligns the student agent to more closely match the real student’s linguistic knowledge. This process involves adapting the student agent’s construction inventory, grammar engine or learning strategies according to a predefined updating scheme.
4. **Give feedback:** the current state of the student profile and the student agent defines which type of feedback is given to the student. The student agent also receives the same feedback, so it can consolidate what it has learned in the current

game through the use of its learning strategies.

5. **Evaluate progress:** after the student's utterance has been processed completely by the language agent, the tutor writes the current interaction data to the student profile, including possible errors that were encountered during parsing, and updates learning problem priority scores. The student has the option to adapt the challenge level at the end of every game, which is then also incorporated by the progress evaluation process.

Only processes 1 and 4 involve the operation of real tutoring strategies. The following section discusses the diagnostics and repairs that are activated by them. The remaining three processes are discussed in Section 9.3.3.

9.3.2 Tutoring strategies to create flow

Tutoring strategies generally serve to provide individualized instruction to students so that their learning process can be optimized. First of all, they should therefore be able to interpret an individual student's actions. According to Mislevy and Yin (2009, p. 254), there are three kinds of data that provide the necessary information to create such an interpretation: "(i) aspects of the situation in which the person is acting, (ii) aspects of the person's action in the situation, and (iii) additional information about the person's history or relationship to the observational situation". The tutor agent also makes use of the same three data kinds; and they translate into the following structures in a tutoring game:

1. the conceptualized situation
2. the student agent after alignment and
3. the updated learning problem queue, together with additional information from the student profile logs.

Once a thorough interpretation has been made, the ultimate goal of the use of tutoring strategies is to guarantee that a student spends most of the time of a tutoring session in his *zone of flow*, where the learning challenge is set in such a way that it does not lead the student either into boredom or anxiety. Yet, the tutor agent alone is not capable of providing a situation of real flow for the student purely based on the student's answers. It needs input from the student to get a good estimate of the appropriate challenge level. This input is provided by the final student process "adapt level", in which the student can either increase or decrease the challenge level or leave it unchanged. The two tutor agent

processes that use this information by means of tutoring strategies are "select situation" and "give feedback". They are discussed in turn in the remainder of this section.

Situation selection strategies

The selection of the situation is the process that has the biggest impact on a student's flow during a tutoring session. Picking the right learning tasks at the right time is an art that every real tutor tries to practice so that students receive individualized training. It is also the process that is the hardest to "simulate" in an intelligent tutoring system, as it involves making intricate decisions, which are captured by tutoring strategies. To make an accurate decision, the tutoring strategies that are active during the select situation process use all remaining components of the tutor agent: the student profile, the student agent and the language agent.

The primary resource that is used by the situation selection tutoring strategies is the learning problem (LP) queue that is kept in the student profile. This queue is ordered according to the priority scores of the learning problems that are in it. Because a tutoring game situation always targets a particular learning problem, it seems thus straightforward to select the learning problem with the highest priority score from the LP queue. Yet, if we take into account the game history and the student's skill level, the highest ranked problem in the queue might not always be the most appropriate one to create a new situation. Selecting the correct learning problem is the first step in situation selection and involves a decision process that makes use of the following diagnostics:

1. **Diagnose-LP-repetition:** Repeating the same target learning problem too many times in a short time span is not beneficial to keep a student in his zone of flow, even if the situations are not exact copies of each other. This diagnostic therefore detects whether the first LP in the queue was repeated too often (according to a threshold) during the last n games (with $n = 10$). The repetition threshold depends on the student level (SL , see Equation 9.4) and is defined as follows:

$$T_r = (1 - SL) * 10 \quad (9.5)$$

The number of target LP occurrences in the last 10 games can be retrieved from the student profile's logs. This number can either be above (or equal) to the threshold T_r or below it. The diagnostic can signal one of these two cases (repetition, no repetition), which are repaired as follows:

- **enqueue-LP:** the target LP is enlisted back at the end of the LP queue and the diagnose-LP-repetition is called again, this time to diagnose the new target LP (in the front of the queue).

- **pop-LP**: when the LP is not repetitive, the highest ranked LP is selected.

This diagnostic can only be called as many times as the number of learning problems in the LP list.

2. **Diagnose-empty-LP-queue– select-LP-from-inventory**: If no LP could be popped from the queue, either because they would lead to repetition or simply because the list is empty, a problem is signaled by this diagnostic. The repair that solves it then selects a random LP from the inventory of predefined LPs (see Figure 9.9), which can be extracted from the problems that the language agent's advanced learning strategies diagnose and repair ¹.

Of course, the above two diagnostics are the first suggestions on how to select an interesting learning problem for a new game. There are certainly more ways to do this that involve taking into account the level that the student has selected (if he is allowed to increase or decrease the challenge level) or inspecting constructions in the student agent's construction inventory that are not yet completely mature.

The selection of a new situation relies on the choice of an appropriate learning problem and the generation of a situation that contains this problem.

Box 9.7 – Situation selection strategies

Yet, a learning problem alone is not enough to instantiate a new language game. We need a situation that can be shown to the student and a question that can be asked, indicating the discourse roles (who is the speaker and who the hearer). The following two steps are used by the tutor agent to instantiate a new situation that fits the selected learning problem, one that applies uniquely to the student agent and one for the student:

1. **Create-new-learning-situation-predicates**: In this step a new situation is created for the current tutoring game based on the learning problem that was selected from the list. The situation consists of a list with conceptualized predicates that is visible by the student agent as well as the language agent. An example of such a situation is present in Figure 9.7a. A situation is generated based on a number of templates that the tutor has in his "tutor toolkit". Templates always start from features that are present in the selected learning problem and add features accordingly.
2. **Transform-predicates**: The second step involves a translation from the above predicate list situation into a real situation that can be shown to the student and will evoke the same conceptualization. A student situation can consist of pictures,

¹ Random selection is not the best for an advanced learner, because this diagnostic will be called often and the tutoring process becomes random. A better solution needs to be brought up in future work.

(green x)
 (apple x)
 (mary y)
 (eat z)
 (eater z y)
 (eaten z x)
 (ongoing z)

(a)



(b)

Figure 9.7 – A new situation for the language and student agents in the form of predicates (a) and in the form of a visual picture (b), which is shown to the real student.

a video clip, a text fragment, etc. An example of a predicate list situation and its translation into a photo is presented in Figure 9.7.

Feedback generation strategies

Feedback tutoring strategies are always based on the communicative success of the tutoring game. A game is successful when the tutor agent could retrieve the topic of the situation without having to consult his learning strategies (i.e. without parsing errors); it fails otherwise. As Section ?? has illustrated, to keep learners in their optimal flow a tutor should foster their intrinsic motivation to learn more. The tutor agent's feedback strategy is therefore to use *descriptive feedback*, in the form of praise or criticism, rather than evaluative feedback. Descriptive feedback helps the learner answer the questions: where am I going, where am I now, and how can I close the gap?

Feedback strategies make use of descriptive feedback that use information about the game's success and the student's level and progression measures (student profile).

Box 9.8 – Feedback strategies

Diagnostics that are used in this tutoring strategy, only function to identify the success of a game, which is also recorded in the student profile:

1. **Diagnose-success:** the language agent did not resort to the meta-level architecture but was able to parse the student's utterance flawlessly. Success is signaled and logged to the student profile.

2. **Diagnose-failure:** the language agent had to call on his learning strategies to interpret the student's utterance and could detect errors that were made and their potential causes. Failure is signaled and logged to the student profile.

Another premise of our tutor agent, apart from the fact that he dislikes evaluative feedback, is his subscription of the situated learning enterprise. Learners that acquire skills and knowledge in a situated learning context are not all the time explicitly told what they did wrong but they learn through experience that consists of trial and error. Also, there are phases in learning when feedback is needed more and quickly absorbed in a situated learning setting. In line with this approach, the repairs that the tutor agent uses do not only rely on the outcome of the game but also check the level of the student to determine whether corrective feedback should be provided:

1. **Create-descriptive-feedback:** both when the game succeeded or failed the tutor can create descriptive feedback that is selected from a list of templates, depending on the game outcome and the game history. Every learning problem that can be diagnosed by the language agent's learning strategies can fill a slot of the prefabricated feedback sentences. Descriptive feedback is only send to the real student, not to the student agent. The student agent is only informed of the communicative success of the game.
2. **Create-corrective-feedback:** after consulting the student agent's constructions (that were not learned in this game) and level measure, the repair decides to provide the correct utterance or not. When the level of the student is low (according to a particular criterium) and the student agent has already acquired the appropriate constructions, the correct utterance is shown to the student. This repair can operate in combination with repair 1.
3. **Provide-explanation:** a linguistic explanation is provided to the student at critical checkpoints in the learning. If the diagnose-recurring-learner-problem diagnostic returns an error occurrence count that is greater than a certain tutor threshold, this repair provides a textbook explanation for the problem in question. Only the real student receives this explanation, whereas the student agent is sent corrective feedback here. This repair can also operate in combination with repair 1.

9.3.3 Remaining tutor agent processes

The remaining tutoring processes of Figure 9.6 do not involve tutoring strategies. They merely carry out basic bookkeeping activities that update the student model, by modifying the student agent's components and extending the student profile as more information about the student is processed. The two processes that this section covers are alignment

CHAPTER 9. FUTURE OUTLOOK

and progress evaluation. The compare utterances process is not covered here because it only involves a straightforward string comparison (student's utterance vs. student agent's utterance).

Alignment

The student agent runs, just as it was the case when there was no real student involved, through its own processes as it communicates with the language agent. It produces the conceptualized situation and parses the feedback of the language agent in consolidation. The same learning strategies that were activated by the student agent to acquire a particular language system on its own are reused here as part of the student model. What is different now, of course, is that the student agent should not only be a good learner and acquire the target language through its learning strategies, but it should also – and more so – mirror the linguistic skills and knowledge of the real student.

To be a good model of the student, the student agent should thus be aligned to the student's utterances so that it can eventually predict them better. As the only source for this alignment is the actual utterance of the student, the most straightforward way is to simply compare the two utterances and base the alignment on their similarities or differences. However, we should also take into account that the language agent could also produce an utterance to express the conceptualized meaning. This utterance is used to check the correctness of the other two. There are thus four scenarios that can occur: (i) the student agent is right, the student is wrong; (ii) the student agent is wrong, the student is right; (iii) both are wrong and (iv) both are right. Yet, if the similarity of the student and the student agent's answers are considered, there is one additional case in which both the answer of the student and the student agent are wrong and also different from each other.

Figure 9.8 shows a toy example of how the student and student agent's utterances can differ with respect to the target tutor utterance, which is in this case the Spanish utterance *cantaba*, 'he/she sang' (imperfective). The main error that was made here (visualized by the dotted-line boxes) concerns a mismatch between the verb class of the verb stem and its suffix: *cantar* belongs to the first verb class and *'ia* is a suffix for the 2nd or 3rd verb class, past imperfective.

The second error is a so-called "bug" because the student clearly knows the correct answer but perhaps made a typographical error or is not yet fully aware of the spelling rules of Spanish. The student agent will never commit bug errors, simply because he always follows the constructions in his inventory and does not make accidental mistakes. When a bug error is found, it is not used for alignment because it might mislead the student agent. Alignment is also skipped when the two utterances are identical, which assumes that the student agent is already aligned to the student, at least in this learning

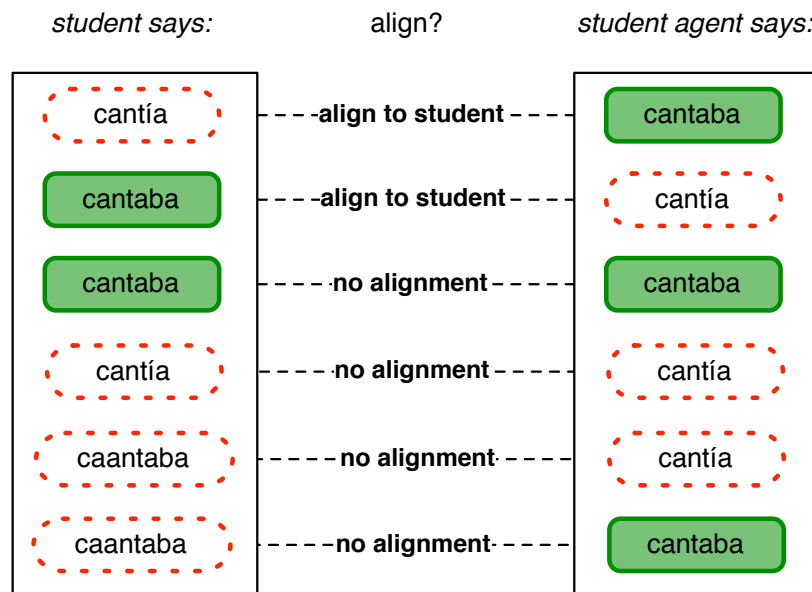


Figure 9.8 – The two inputs that the tutor receives – that of the real student and that of the student agent – are compared and alignment happens accordingly. The unfilled boxes (dotted lines) signal inconsistencies with the tutor answer, the green filled boxes are correct answers.

situation. This leaves us with only two cases that require alignment: when the student committed a real error while the student agent was right and reversely. The alignment works differently in these two cases:

1. The student's utterance can be diagnosed by the language agent as erroneous while the student agent's utterance is correct when the student agent has learned too quickly in the current interaction (using its learning strategies) or the correct form was learned earlier in the tutoring session but has not yet been consolidated by the real student. Because the student agent should mirror the real student, the constructions that were used to produce the correct answer should be revised according to the following scheme:
 - If they were learned in the current game, the student agent's construction inventory is simply reverted to its previous state.
 - If there was no learning in the current game, the real student's utterance is parsed and the student agent's learning strategies create constructions that learn the form-meaning mapping given the current situation. Of course, if the utterance could be parsed and its parsed meaning corresponded to the conceptualized meaning of the situation, no new constructions have to be learned but the success scores of the constructions that produced the correct answer (the *aba*-construction) needs to be decreased.

2. The opposite case occurs when the real student learns faster than the student agent. The student agent should thus also acquire the constructions that the student knows. This learning boost can happen in two alternative ways, of which I prefer the first one:

- The student agent parses the correct solution and uses its learning strategies to acquire the needed constructions.
- The constructions that the language agent used to produce the utterance are combined with the student agent's constructions and potential differences are restored.

Progress evaluation

To define the impact of a game and the provided feedback, whether positive or negative, as well as the learning progress of the student, we need to get an understanding of the how well the student agent "knows" the constructions that were affected by the feedback. This effect becomes visible after the student agent has gone through the consolidation process, which updates the confidence scores of all constructions that were used to construct the utterance. Remember, alignment takes place when the student's utterance did not contain a bug and it differed from the student agent's utterance. Alignment can reset confidence scores of existing constructions back to a default score (0.5) or a minimum (0.0) when the student did not have the knowledge that the student agent predicted. In the opposite scenario, when the student agent's construction inventory was boosted with the target language agent's constructions, it sets the confidence scores of these constructions to a maximum (1.0).

Progress evaluation estimates how well the student agent masters the constructions that were affected by the feedback provided in the current game.

Box 9.9 – Progress evaluation

The student profile contains a feedback impact measure, which estimates the effect of the feedback that was given to the student (agent). The measure is calculated as the average of the confidence scores of the constructions that are affected by the feedback. Feedback-affected constructions are those constructions that share the same FCG features as the learning problem that is currently handled. To evaluate a student's progress, the feedback impact should be evaluated according to a certain threshold, which I will refer to as the learning-effectiveness threshold. I define this learning-effectiveness threshold here as a parameter that is relative to the initial confidence score that newly added constructions receive ($cs_{init} = 0.5$):

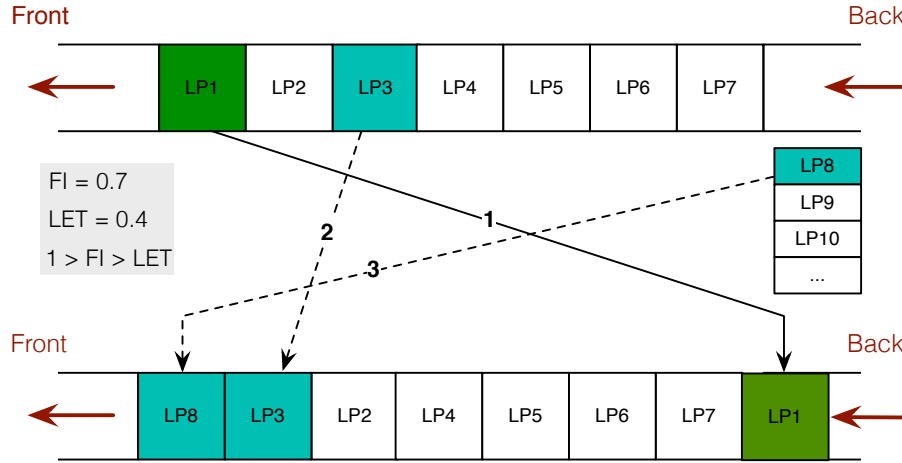


Figure 9.9 – The evaluation of a student's progression results in an updated LP list. The target LP (LP1) is enlisted again at the end of the list, because the calculated feedback impact score (0.7) was found above the learning effectiveness threshold but below the maximum score of 1.0. Two diagnosed LPs were pushed to the front of the list: LP8, a newly added problem, and LP3, which was already part of the LP list.

$$T_{le} = cs_{init} - 0.1 \quad (9.6)$$

There are three scenarios that can occur when the learning effectiveness is measured (see Figure 9.9):

1. **Low learning effectiveness** ($FI \leq T_{le}$): If the FI score is smaller or equal to the LE threshold, the student has not learned enough to be confident to use the constructions needed in the current learning situation. The target learning problem's priority score is increased.
2. **Medium learning effectiveness** ($1 > FI > T_{le}$): The FI score is above the effectiveness threshold, which indicates that the affected constructions still have some open feature values or need more positive reinforcement. The target learning problem is appended at the back of the LP queue.
3. **High learning effectiveness** ($FI = 1$): the student manages all affected constructions perfectly (all have a confidence score of 1). The target learning problem is removed from the LP list.

It can also happen that new learning problems appear during one tutoring game, which are caught during the language agent's parsing process. They are also pushed to the front of the list, similarly as the problems that are already waiting in the LP list. Yet, because this is only a design plan, the real implementation might require some additional

changes to work in practice.

9.4 The Spanish verb tutor

Unfortunately, the design of the student model and the tutoring strategies as described above has not yet been implemented in an interactive language game format. Although the plan is relatively clear now, its implementation will tell whether it is successful or not. To give you an idea of how you could interact with the Spanish verb tutor, I illustrate here a few example games in a mockup screen that resembles the planned interface.

In the first example (Figure 9.10a), the student is the hearer and receives a sentence ("Ayer, recogieron muchas peras del árbol") that he has to link to one of the pictures that is given. Such a comprehension task is thus always a discrimination game, with two or more possible situations that can be linked to the tutor's utterance. In Figure 9.10a, the student is asked to click on one of the two pictures that he believes expresses the meaning of *recogieron* 'they picked'. The only difference between these two pictures is the number of participants in the picking action, which is reflected in the person feature of the verb form: 3SG vs 3PL. The student has the option to skip this game when he does not like it (it can be too difficult or too easy). An alternative comprehension task arranges situations on a time line and asks the student to match a particular verb form with one of the situations on the time line (Figure 9.10b).

Production will in the first stage be constrained so that the student can merely select a possible verb form from a dropdown menu rather than insert free text so that the input can be controlled for more. Figure 9.11 shows how the student is asked to complete a sentence that describes the picture on the left. Now, because the student needs to choose from a list of conjugated verb forms, it is sometimes easier to start from the infinitive to make a choice. An additional button in the interface provides the option of revealing the infinitive. The three options that are provided includes one ungrammatical form: *recoge* (< *recoger* 'to pick'). The remaining forms are the indicative and subjunctive present, both in 3rd person singular.

Feedback is provided every time the student pointed to an event in comprehension or produced an event description in production. Negative feedback corrects the student and optionally gives an explanation on the source of the error, as has been illustrated in Figure 9.12a. When the student has answered positively, the tutor agent praises him and asks him to do one more exercise on the same problem with one feature different (see Figure 9.12b). At this moment in a game, the student also gets the opportunity to inspect his progress or to adjust the skill level of the games.

Which picture describes the following sentence?

Ayer, **recogieron** muchas peras del árbol.



Click on the picture to answer this question

Skip this game

(a) Discriminating number and person

Can you link the conjugated verb form to the situation?

ha recogido




Check the corresponding box on the time line

Skip this game

(b) Discriminating tense

Figure 9.10 – A comprehension task can either consist of two events (pictures), from which the student needs to choose the one most closely matching the utterance that he has parsed (a), or it contains a timeline with events laid out on it (b).

Can you complete this sentence by selecting the fitting verb form?




¡Mira! El chico recoje
recoge
recoja una pera

Reveal the infinitive Skip this game

Figure 9.11 – The student has selected an event from a situation with multiple events and formulates a verb form that describes this event. Instead of entering free text, he is restricted to use the three options that the system provides.

You answered: El chico **recoge** una pera




Unfortunately, that's **not correct...**
To maintain the same pronunciation,
the stem's last letter -g- changes into -j-:

El chico **recoje** una pera

Adjust the level Inspect my progress Continue

(a) Negative feedback

You answered: El chico **recoje** una pera



You almost sound like a native speaker now!
Can you also do the following one?

La semana pasada, recogimos
recojimos peras del huerto.

Adjust the level Inspect my progress Skip this game

(b) Positive feedback

Figure 9.12 – The student receives feedback of the tutor agent, which can be negative (a) or positive (b).

9.5 Conclusion

A tutor agent is formed by combining the expertise of the language agent and the stepwise learning strategies of the student agent into an interactive language tutoring system that can receive input from a student and proactively create new learning situations and generate feedback by calling upon a pre-specified set of tutoring strategies. The same meta-level architecture reappears here, with learning problems that manifest itself during the interactions (diagnosed by the language agent or student agent) and are tackled by repairs that carry out actions that influence the learning path of the student. It is through the plugging in of tutoring strategies that the learning process becomes personal and adapted to the student.

Still, because a lot of time and effort was put into the development of the necessary basics of the tutor agent (the processing and the learning components), a working implementation is currently missing. The first steps in completing the full tutor agent would thus be to implement a user interface through which the tutor agent can interact with the student and to formalize the processes of Figure 9.6 that a tutor agent runs through. Further, diagnostics and repairs for the tutoring strategies and a data structure for the student profile follow in the next steps of the planned operationalization. Although the mock-up interface has shown how the student can only select verb forms and not really create them by himself, a second prototype phase will experiment with free form input by the student. Of course, once the prototype can be tested in a pilot study it needs to be incorporated in a web service that students can access on their smartphones, tablets or laptops from anywhere, at any time. A login would be required to track an individual student over multiple learning sessions.

Part IV

Conclusions

Chapter 10

Conclusions

Nevertheless, there are still a lot of open problems. Many systems focus on single aspects of language learning. What is needed is a more integrated and comprehensive approach which supports also semantics, pragmatics, cultural knowledge and social abilities, using different technologies which are tailored for the training of specific skills. (Gamper & Knapp, 2002, p. 339)

Although this quote is now more than ten years old, the "open problems" that Gamper and Knapp mention have still not been solved. Yet, since the year 2000 language learning software has shifted from stand-alone software packages to web services that require a login, thereby opening the doors for real interactive mobile learning. A thorough integration of the semantics, pragmatics and cultural knowledge a student needs has not yet been achieved and I believe that *situated learning in open-ended environments* is the kind of learning that future language learning application developers should strive for. This dissertation was born out of ideas on "The future of learning", a series of workshops and books led by Luc Steels (2003, 2004, 2012) in which the idea of self-motivated learning and the theory of flow were central themes. Of course, building a piece of language learning software requires technical skills that unite techniques from the domain of expert systems (knowledge about typical mistakes, questions, answers), intelligent tutoring systems (student and tutor modules), user modelling and adaptive systems (adapt content according to a user's steps) and natural language processing (analyse open student linguistic production). This final chapter recaptures how these techniques were employed to lay the foundations of a new sort of language tutoring system in which a proficient language agent (expert) and a student agent (learner) interact.

10.1 Achievements

This dissertation achieved two main objectives (defined in Chapter 1):

1. The first achievement is a comprehensive model of the computational mechanisms behind Spanish verb conjugation. This model can formalize any verb conjugation in the Spanish language (including neologisms) thanks to an innovation in the segmentation algorithm of Fluid Construction Grammar and the introduction of phonological constructions that work on the phonetic profile of a verb stem or ending. Moreover, flexible *processing* is guaranteed thanks to a set of flexibility strategies that are used to catch and correct verb form errors made by learners.
2. The second achievement is to not only have a ready-made model of Spanish verb conjugation but also to be able to acquire the language system on a gradual basis through interactions with a proficient speaker. This *learning* process is established through the use of learning strategies that have been created specifically for the L2 acquisition of Spanish morphology, tense, aspect and mood.

These two achievements have paved the way for a Spanish verb tutor, which relies on flexibility strategies to understand a student's utterances and learning strategies that reconstruct the learning process and the acquired (partial) constructions to build a functional model of the student's knowledge. The example of the Colour tutoring game, which was based on elaborate research on the emergence and the evolution of a lexicon for colour words (PhD dissertations of Tony Belpaeme (2002) and Joris Bleys (2011)), has shown the first potential of using language games for tutoring purposes. The remainder of this section discusses individual achievements of the two objectives. Future elaborations of a tutoring language game for Spanish are explored in Section 10.2.

10.1.1 Reconstructing Spanish verb conjugation

The first objective of this thesis is to gather all information needed to design and build a construction inventory and a grammar engine for Spanish verbs to be used by a proficient language agent. Spanish verb conjugation is not a straightforward task due to its rich morphology and the high number of semi-regular verb conjugations (irregular in some parts of the paradigm). Moreover, learning when to use the past perfect and the past imperfect correctly is perhaps the most difficult part of learning Spanish as an adult. The difference between *cantía* 'he sang.IMPF' and *cantaba* 'he sang.PF' simply does not come very natural. The following paragraphs summarise the main achievements of this dissertation with respect to the creation of a formal grammar for the conjugation of any verb in Spanish.

Constructions for phonological changes Fluid Construction Grammar (FCG) is normally used to formalize an inventory of lexical, functional, phrasal and morphological constructions of a particular language system. Constructions that focus on sound changes under certain conditions had never been formalized in FCG. This new type of constructions works on a special constructional feature **phon-cat**, which contains a stem's segmentation pattern (onset-nucleus-coda), stress information and suffix vowels. Phonological constructions can thus still change a certain form during processing (stem or ending) when certain syntactic conditions are met (e.g. 1sg past perfect). A new special operator was introduced to allow for a more general application of certain constructions: the **=r** operator. This operator introduces regular expressions for string matching: e.g. **"*c"** matches the stem's coda **"rc"** or **"c"**.

Flexibly parsing students' errors Apart from a construction inventory and a grammar engine that are specialized for Spanish verb conjugation, I developed a set of flexibility strategies that contribute to the language proficiency of the language agent. Similar to a native speaker, the language agent can spot the smallest deviation in his interlocutor's utterances and restore the error to successfully process the full utterance. Flexibility strategies always trigger on a problem that occurs, whether in parsing or production (the utterance is too difficult for the student to understand). Because the manifested problem can be caught by a diagnostic and solved by a repair, regular processing can continue without any visible interruptions.

The language agent can also correct a student's errors by means a repair that returns the corrected form, optionally including the source of the error. The corrections are evaluated on a corpus of learner errors of English students learning Spanish at an undergraduate level (SPLLOC). For the evaluation, I selected all verb form errors and extracted the original error along with the human annotated correction. The language agent was asked to parse every individual error and suggest one correction. A correction is assumed to be accurate when it is equal to the human's correction. Results have shown that the total correction accuracy is equal to 60% but when context is taken into account (not only a verb form is parsed but a situation is linked to it) it mounts to 79%.

Bootstrapping a large grammar To provide the language agent with a realistically sized construction inventory an algorithm was developed to bootstrap a grammar based on a list of infinitives. A decision tree classifier is used to sort infinitives according to a particular verb type they belong to. Once a verb type could be assigned, the corresponding constructions are automatically added to the grammar. When a verb type already existed in the grammar, the only construction that needs to be added is a lexical one for the lemma. The language agent's default grammar was fed with the 600 most common verbs in Spanish and resulted in a construction inventory of around 3000 constructions. The effectiveness of this grammar is evaluated on a large set of conjugations of the 600 input infinitives (> 11 000 forms). Every conjugated verb form of this corpus was parsed and reproduced to estimate the grammar's accuracy. Overall, the accuracy was

100% except for the parsing of ambiguous verb forms that could belong to any of two conjugational paradigms: e.g. *presentir* vs. *presentar*.

10.1.2 Learning strategies for Spanish verbs

The grammar bootstrapping algorithm was a shortcut to quickly create a large-scale grammar that yields very accurate processing results. Yet, to track a student's constructional knowledge over time, a more gradual approach is needed. A student agent, contrary to a language agent thus starts with an empty construction inventory that is gradually built up when more utterances are processed. Learning strategies diagnose problems that occur during processing due to a lack of proficiency. A set of predefined repairs solve learning problems by adding a particular construction that was missing or adapting an existing construction to be more usable in the given context. Apart from formalizing basic learning strategies to deal with unknown verb stems or endings, the most significant achievements are the learning of grammatical constructions for tense and aspect and the learning of verb classes. They are both treated in the following two subsections.

Learning grammatical constructions Grammatical constructions are more general constructions that can be used in a multitude of processes: e.g. all past perfect verb form conjugations, all 3rd person singular forms, etc. They are not learned in one step but multiple learning instances can trigger the learning of a grammatical construction. The main innovation of a grammatical construction is that it links the arguments of the stem construction with those found in the suffix. The grammatical meaning expressed by one specific suffix is thus externalized by a grammatical construction that can be used by multiple suffixes that express the same meaning with a different form.

Learning verb classes The three verb classes in Spanish are not given to the student agent as part of the learning strategies but have to be learned based on the verb forms that are processed in parsing or production. Verb classes become visible in the verb endings, which differ for the same grammatical meaning based on the class of the verb stem they are combined with. Yet, some endings are shared between two classes, which makes it less transparent for the student agent to detect the class of a certain verb. A classification strategy was presented as part of the learning strategy for verb classes that leads to complete communicative success with a minimal number of three verb categories. Yet, a student agent's grammar can contain more than three categories when an ungrammatical verb form is produced by the student agent and the suffix that was used did not yet belong to one of the three existing verb classes. In such cases, a fourth verb class is created. An additional learning operator could later merge two verb classes when all its suffixes are shared but this option is currently not implemented.

10.2 Future work

Once the two basic components of the tutor agent are robust enough to function as the expert model and student model, its operationalization lies within reach. The development of tutoring strategies is the first step to complete a tutor agent that can adapt its learning materials to the level of the student that is being tutored. A tutoring strategy is defined as a plan to teach a student certain learning materials so that the student's learning objectives are maximized. First suggestions on the data structures these tutoring strategies work on (learning problem list, student profile) are made in this dissertation. The basic script of the discriminative language game was introduced for the language agent - student agent interactions. This script will be reused for the tutor agent - student interactions, with situations in which a particular event conceptualization needs to be expressed in contrast with other events. Tutoring strategies assist in selecting contrastive events for a new learning situation so that the student is optimally challenged. When particular grammatical features still form an obstacle (e.g. aspect) or stem changes of a particular verb type have not yet been mastered they will have a high priority in the learning problem list. One tutoring strategy might go through the list and select a learning problem for the next interaction. Based on this learning problem, a new contrastive situation is created.

The tutoring strategies module follows the same basic principles of diagnostics and repairs as the flexibility strategies and the learning strategies. Yet, they do not only need the problem list to work on (similar to the problem-based learning approach in the student and the language agents) but a library with possible situations or event constellations needs to be provided. This library is vital to generate new learning situations that simulate situated learning and keep a student's learning challenge up to an adequate level. For this library audiovisual material needs to be selected and annotated with features that match with constructional features that can be learned by a student agent's learning strategies. Moreover, a second function of the tutoring strategies module is to create constructive feedback (descriptive) that helps the student in further understanding the source of his mistake and give him an idea of his strengths and weaknesses. Explanations that elicit the grammatical contrast present between two events in a given situation are sometimes needed to really offer comprehensive error analysis to certain students.

Yet, tutoring strategies are not the only contributor to the overall learning success of future students that interact with the Spanish tutor agent. To really influence the experience of flow students should be allowed to actively change the challenge level of the current game situation. This empowering action is also advantageous for the tutor agent because it offers an insight into the real level of the student, which might have been misjudged due to a lack of information. Apart from changing the language proficiency level, another possibility that should be investigated is whether the student can be allowed to also select a new learning situation by himself. First of a predefined set of situations

CHAPTER 10. CONCLUSIONS

and eventually by making a picture of a certain real-life situation and using that for a next interaction with the tutor agent. This last scenario points to the real goal of tutoring strategies: the creation of relevant learning situations for every individual student.

Perhaps the biggest challenge and a research topic on its own is the alignment of the constructions mastered by the student agent and the linguistic knowledge of the real student. This alignment process is not dependent on a tutoring strategy but remains the same for any student that interacts with the system. It should happen after every single interaction and uses the following information (in a production task): the situation, the topic event, the student's utterance, the student agent's utterance, the (potential) correction by the language agent and the student agent's construction inventory (including information on what was learned in the current interaction). The error analysis carried out by the flexibility strategies can be useful to pinpoint the exact constructional features that are still insufficiently covered by the grammar. How exactly a student agent's construction inventory should be populated with constructions in the initial state of the tutoring session and how large the effect of the alignment operation is at the end of a certain game are interesting research questions that deserve to be explored in future research.

Linguists and language educators are now beginning to understand that students learn a language more efficiently when constructions are presented to them rather than individual words in a vocabulary list that are separated from abstract grammar rules. A subfield specifically focused on language pedagogy is emerging within the Construction Grammar community (see for instance the *Constructional Approaches to Language Pedagogy* conference, 8 - 9 November 2013, Brussels). This emerging strand of research needs to be taken into account when developing a set of tutoring strategies for verb teaching as well as in the creation of new learning situations. Also, if we understand better how humans learn and unlearn constructions, the alignment process will be easier to implement.

The first prototype of the Spanish verb tutor will be build on the same platform as the Colour Tutoring Game that has been presented in this dissertation. Yet, the real potential lies of course in a web service application that can be accessed from anywhere at any time by the student. Through a simple user account every individual student can be traced and log information and progress measures across tutoring sessions are kept in the student profile module of the tutor agent. Starting with controllable multiple choice exercises, fill the gap tasks allow slightly more input from the student but should be controlled for the correct use of diacritics in Spanish. Future extensions to use recorded speech segments to describe events or a speaking task that lets a student record his answer should be envisioned in long-term future plans. The real future lies of course in a language tutor that goes beyond the conjugation of verbs and stretches over different language subsystems of the target language (vocabulary use, adjective declension, noun phrase patterns, prepositions, politeness forms, etc.). Yet, I believe that effectively incorporating more language systems and multiple target languages can

only be achieved when the tutoring game can be reversed to train the language agent to become a competent language user (similar to the Colour Tutoring Game human teaching scenario).

10.3 Final remarks

As could be shown in this dissertation, the development of the basic building blocks of a language tutoring system requires a lot of time and effort, which did not yet result in a working prototype. Yet, both the competent language agent as the student agent are fundamental aspects that need to be stable before any tutor-student interaction can be envisioned. Whereas the language agent is responsible for the interaction with the student (exclusively using the target language), error diagnosis and analysis and modelling a competent language user, the student agent is the heart of the active student model that can trace a student's utterances and adapts its grammar by means of the learning strategies that it contains. It is only when these two components are into place that a new type of artificial language tutor can arise.

Bibliography

- Abeillé, A. (1992). A lexicalized tree adjoining grammar for french and its relevance to language teaching. In *Intelligent tutoring systems for foreign language learning* (pp. 65–87). Springer.
- Achard, M. (2008). Teaching construal. In P. Robinson & N. Ellis (Eds.), *Handbook of cognitive linguistics and second language acquisition* (pp. 432–456). Mahwah, NJ: Lawrence Erlbaum Associates.
- Ackermann, E. K. (2004). Constructing knowledge and transforming the world. In M. Tokoro & L. Steels (Eds.), *A learning zone of one's own* (Chap. 2, pp. 15–36). IOS Press.
- Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems : example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2), 105–154.
- Amaral, L. & Meurers, D. (2007). Conceptualizing student models for ICALL. *User Modeling 2007*, 340–344.
- Amaral, L. & Meurers, D. (2008, October). Little things with big effects: On the identification and interpretation of tokens for error diagnosis in ICALL. *CALICO Journal*, 26(3), 580–591.
- Anderson, J., Corbett, A., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *The journal of the learning sciences*, 4(2), 167–207.
- Arroyo, I., Cooper, D. G., Burleson, W., Woolf, B. P., Muldner, K., & Christopherson, R. (2009). Emotion sensors go to school. In *Proceeding of the 2009 conference on artificial intelligence in education, july 6th-10th, brighton, uk, ios press* (pp. 17–24).
- Baker, R. S. J., D'Mello, S. K., Rodrigo, M. M. T., & Graesser, A. C. (2010). Better to be frustrated than bored: the incidence, persistence, and impact of learners' cognitive–affective states during interactions with three different computer-based learning environments. *International Journal of Human-Computer Studies*, 68(4), 223–241.
- Basterrechea, E. & Rello, L. (2010). *El verbo en español*. Molino de Ideas.
- Batali, J. (1983, February). *Computational Introspection* (tech. rep. No. AI Memo No. 701).

BIBLIOGRAPHY

- Beck, J., Chang, K., Mostow, J., & Corbett, A. (2008). Does help help? introducing the bayesian evaluation and assessment methodology. In *Intelligent tutoring systems* (pp. 383–394). Springer.
- Beer, M. D. & Whatley, J. (2002). A multi-agent architecture to support synchronous collaborative learning in an international environment. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems: part 1* (pp. 505–506). ACM.
- Beesley, K. R. & Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques. *CSLI*.
- Bergen, B. K. (2003). Towards morphology and agreement in Embodied Construction Grammar. URL www2.hawaii.edu/bergen/papers/ECGmorph.pdf.
- Bergen, B. K. & Chang, N. (2005). Embodied construction grammar in simulation-based language understanding. In J. Östman & M. Fried (Eds.), *Construction grammar(s): cognitive and cross-language dimensions* (27). Embodied construction grammar in simulation-based language understanding. Amsterdam: John Benjamins.
- Beuls, K. (2011). Construction sets and unmarked forms: a case study for Hungarian verbal agreement. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 237–264). Amsterdam: John Benjamins.
- Beuls, K. (2012). Handling scope in Fluid Construction Grammar: a case study for Spanish modals. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- Beuls, K., Gerasymova, K., & van Trijp, R. (2010). Situated learning through the use of language games: solving the hungarian agreement puzzle. In *Proceedings of benelearn 2010*. Leuven.
- Beuls, K., van Trijp, R., & Wellens, P. (2012). Diagnostics and Repairs in Fluid Construction Grammar. In L. Steels & M. Hild (Eds.), *Language Grounding in Robots*. New York: Springer.
- Beuls, K. & Wellens, P. (2010). Linking constructions and categories: a case study for hungarian object agreement. In *Book of abstracts of the sixth international conference on construction grammar* (44–45).
- Bishop, C. (2006). *Pattern recognition and machine learning*. Information Science and Statistics. Springer.
- Bleys, J., Stadler, K., & De Beule, J. (2011). Search in linguistic processing. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.
- Bloom, B. (1984). The 2 sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13, 3–16.
- Bodnar, S. (2012). Enhancing motivation in intelligent computer-assisted language learning. In *Eurosla22 22nd annual conference of the european second language association* (p. 210).

- Boicu, M., Tecuci, G., Stanescu, B., Marcu, D., Barbulescu, M., & Boicu, C. (2004). Design principles for learning agents. In *Proceedings of aaai-2004 workshop on intelligent agent architectures: combining the strengths of software engineering and cognitive systems* (pp. 26–33).
- Booij, G. (2010, September). *Construction Morphology*. OUP Oxford.
- Bosque, I. & Demonte, V. (1999). *Gramática descriptiva de la lengua española*. Espasa: Real Academia Española, Colección Nebrija y Bello.
- Bourdeau, J. & Grandbastien, M. (2010). Modeling tutoring knowledge. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 123–143). Springer.
- Brocklebank, C. P. (1998). *An experiment in developing a prototype intelligent teaching system from a parser written in prolog* (Master's thesis, UMIST, Manchester).
- Bruce, B. C. (n.d.). The history of instructional technology.
- Bull, W. (1965). *Spanish for teachers: applied linguistics*. R.E. Krieger Pub. Co.
- Burke, D. L. (1996). Multi-year teacher/student relationships are a long-overdue arrangement. *Phi Delta Kappan*, 77(5), 360–61.
- Capuano, N., Marsella, M., & Salerno, S. (2000). Abits: an agent based intelligent tutoring system for distance learning. In *Proceedings of the international workshop on adaptive and intelligent web-based education systems, its*.
- Carbonell, J. (1970a). Ai in cai: an artificial intelligence approach to computer aided instruction. *IEEE Transactions on Man-Machine Systems*, 11, 190–202.
- Carbonell, J. (1970b). *Mixed-initiative man-computer instructional dialogue* (Doctoral dissertation, MIT, Cambridge, MA).
- Carr, B. & Goldstein, I. P. (1977). *Overlaps: a theory of modeling for computer-aided instruction* (Technical Report AI Lab Memo No. 406). Massachusetts Institute of Technology.
- Cho, E. H. & Larke, P. J. (2010, December). Repair Strategies Usage of Primary Elementary ESL Students: Implications for ESL Teachers. *The Electronic Journal for English as a Second Language*, 14, 1–18.
- Chomsky, N. & Halle, M. (1968). The sound pattern of english.
- Clancey, W. J. (1979). *Transfer of rule-based expertise through a tutorial dialogue* (Doctoral dissertation, Stanford University, Stanford, CA).
- Clancey, W. J. (1987). *Knowledge-based tutoring: the guidon program*. Cambridge, MA: MIT Press.
- Comeau, L., Genesee, F., & Mendelson, M. (2007, January). Bilingual children's repairs of breakdowns in communication. *Journal of Child Language*, 34(01), 159.
- Cook, R. & Kay, J. (1994). *The justified user model: a viewable, explained user model*. Basser Department of Computer Science, University of Sydney.
- Copestake, A. & Flickinger, D. (2000). An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd international conference on language resources and evaluation (LREC 2000)* (pp. 591–600).

BIBLIOGRAPHY

- Corbett, G. G. (2003, August). Agreement: the range of the phenomenon and the principles of the Surrey Database of Agreement. *Transactions of the philological society*, 101(2), 155–202.
- Corder, S. P. (1974). *Error analysis*. Oxford: Oxford University Press.
- Cormier, D. & Siemens, G. (2010, August). "THROUGH THE OPEN DOOR: OPEN COURSES AS RESEARCH, LEARNING, AND ENGAGEMENT", 1–7.
- Cornillie, F., Thorne, S. L., & Desmet, P. (2012, August). Recall special issue: digital games for language learning: challenges and opportunities. *ReCALL*, 24, 243–256. doi:10.1017/S0958344012000134. eprint: http://journals.cambridge.org/article_S0958344012000134
- Crawford, J. (1987). *EMYCIN: An Expert System Shell*. Technical report (University of Sydney. Basser Dept. of Computer Science). Basser Department of Computer Science, University of Sydney.
- Creutz, M. & Lagus, K. (2002, May). Unsupervised Discovery of Morphemes. *arXiv.org, cs.CL*.
- Croft, W. (2001). *Radical construction grammar: syntactic theory in typological perspective*. Oxford: Oxford UP.
- Croft, W. & Cruse, D. A. (2004). *Cognitive linguistics*. Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.
- Crowder, N. A. (1964). On the difference between linear and intrinsic programing. In A. G. Grazia & D. A. Sohn (Eds.), *Programs, teachers, and machines* (pp. 77–85). New York, NY: Bantam Books.
- Csikszentmihalyi, M. (1978). *Beyond boredom and anxiety: experiencing flow in work and play*. Cambridge: Cambridge University Press.
- Daelemans, W. & Van den Bosch, A. (2005). *Memory-based language processing*. Studies in Natural Language Processing. Cambridge: Cambridge University Press.
- Declerck, R. (1991). *Tense in English: Its Structure and Use in Discourse*. Germanic linguistics. Routledge.
- Della Fave, A. & Bassi, M. (2003). Italian adolescents and leisure: the role of engagement and optimal experience. *New Directions for Child and Adolescent Development*, 2003(99), 79–94.
- Delmonte, R. (2002). Feedback generation and linguistic knowledge in slim automatic tutor. *ReCALL*, 14(2), 209–234.
- Delmonte, R. (2003). Linguistic knowledge and reasoning for error diagnosis and feedback generation. *Calico Journal*, 20(3), 513–532.
- d’Inverno, M. (2012, December). Motivation and praise. Presentation on The future of learning workshop in Casteldelfells.
- Dornbusch, S. M., Elworth, J. T., & Ritter, P. L. (1988). Parental reaction to grades: a field test of the over justification approach. *Unpublished manuscript, Stanford University*.

- Egbert, M. (1998). Miscommunication in language proficiency interviews of first-year german students: a comparison with natural conversation. In R. Young & W. He (Eds.), *Talking and testing: discourse approaches to the assessment of oral proficiency* (pp. 147–169). Amsterdam: John Benjamins.
- Ellis, R. (1997). *Second Language Acquisition*. E.L.T. Selections: Articles from the Journal English Language Teaching. OUP Oxford.
- Ferreira, A., Moore, J. D., & Mellish, C. (2007). A Study of Feedback Strategies in Foreign Language Classrooms and Tutorials with Implications for Intelligent Computer-Assisted Language Learning Systems. *International Journal of Artificial Intelligence in Education*, 17, 389–422.
- Feurman, K., Marshall, C., Newman, D., & Rypa, M. (1987). The calle project. *Calico Journal*, 4(3), 25–34.
- Feurzeig, W., of Computing Activities, N. S. F. (O., Bolt, B., & Newman, i. (1969). *Programming-languages as a conceptual framework for teaching mathematics: final report on the first fifteen months of the logo project*. BBN report. National Science Foundation, Office of Computing Activities.
- Fischer, R. (2013, January). A Conceptual Overview of the History of the CALICO Journal: The Phases of CALL. *CALICO Journal*, 30(1), 1–9.
- Foth, K., Menzel, W., & Schröder, I. (2005). Robust parsing with weighted constraints. *Natural Language Engineering*, 11(1), 1–25.
- Fried, M. & Östman, J.-O. (2004). Construction Grammar: A Thumbnail Sketch. In M. Fried & J.-O. Östman (Eds.), *Construction grammar in a cross-language perspective* (pp. 11–86). Amsterdam: John Benjamins.
- Gamper, J. & Knapp, J. (2002). A Review of Intelligent CALL systems. *Computer Assisted Language Learning*, 15(4), 329–343.
- Gerasymova, K. (2012). Expressing grammatical meaning with morphology: a case study for Russian aspect. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- Gerasymova, K. & Spranger, M. (2010). Acquisition of grammar in autonomous artificial systems. In *Proceedings of the 19th european conference on artificial intelligence (ecai-2010)*.
- Gerasymova, K., Spranger, M., & Beuls, K. (2012). A language strategy for aspect: encoding aktionsarten through morphology. In L. Steels (Ed.), *Experiments in cultural language evolution* (pp. 257–276). John Benjamins Amsterdam.
- Godwin-Jones, R. (2011). Emerging Technologies: Mobile Apps for Language Learning. *Language Learning & Technology*, 15(2), 2–11.
- Goldberg, A. E. (1995). *A Construction Grammar Approach to Argument Structure*. Chicago: Chicago UP.
- Goldberg, A. E. (2006). *Constructions at work: the nature of generalization in language*. Oxford: Oxford University Press.
- Goldberg, A. E. & Suttle, L. (2010). Construction grammar. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(4), 468–477. doi:[10.1002/wcs.22](https://doi.org/10.1002/wcs.22)

BIBLIOGRAPHY

- Goodkovsky, V. et al. (1997). Pop class intelligent tutoring systems: shell, toolkit & design technology. *Moscow State Institute for Physics and Engineering*.
- Gottfried, A. E., Fleming, J. S., & Gottfried, A. W. (1994). Role of parental motivational practices in children's academic intrinsic motivation and achievement. *Journal of Educational Psychology*, 86(1), 104.
- Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005, November). AutoTutor: an intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4).
- Graesser, A. C. & Person, N. K. (1994). Question asking during tutoring. *American Educational Research Journal*, 31, 104–137.
- Hamel, M.-J. (1996). Nlp tools for nlp in call for error diagnosis. *Journal of the CAAL*, 18(2), 125–141.
- Hart, R. (1981). Language study and the plato system. *Studies in Language Learning*, 3(1), 1–24.
- Heift, T. (1998). *Designed intelligence: a language teacher model* (Doctoral dissertation, Simon Fraser University).
- Heift, T. (2001). Error-specific and individualised feedback in a web-based language tutoring system: do they read it? *ReCALL*, 13(1), 99–109.
- Heift, T. (2002). Learner control and error correction in icall: browsers, peekers, and adamants. *Calico Journal*, 19(2), 295–313.
- Heift, T. (2003). Multiple learner errors and meaningful feedback: a challenge for icall systems. *CALICO journal*, 20(3), 533–548.
- Heift, T. & Nicholson, D. (2000). Enhanced server logs for intelligent, adaptive web-based systems. In *Proceedings of the workshop on adaptive and intelligent web-based educational systems*, its (pp. 23–28).
- Heift, T. & Nicholson, D. (2001). Web delivery of adaptive and interactive language tutoring. *International Journal of Artificial Intelligence in Education*, 12(4), 310–325.
- Heift, T. & Schulze, M. (2003a). Error analysis and error correction. *Special Issue of the CALICO Journal*, 20(3).
- Heift, T. & Schulze, M. (2003b). Student Modeling and ab initio Language Learning. *System*, 31(4), 519–535.
- Heift, T. & Schulze, M. (2007). *Errors and intelligence in computer-assisted language learning: parsers and pedagogues*. New York, London: Routledge.
- Heinecke, J., Kunze, J., Menzel, W., & Schröder, I. (1998). Eliminitive parsing with graded constraints. In *Proceedings of the 36th annual meeting of the association for computational linguistics and 17th international conference on computational linguistics-volume 1* (pp. 526–530). Association for Computational Linguistics.
- Hmelo-Silver, C. E. (2004). Problem-based learning: what and how do students learn? *Educational Psychology Review*, 16(3), 235–266.

- Höfer, S. (2012). Complex declension systems and morphology in Fluid Construction Grammar: a case study of Polish. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- Hospers, M., Kroezen, E., Nijholt, A., op den Akker, R., & Heylen, D. (2003). An agent-based intelligent tutoring system for nurse education. In *Applications of intelligent agents in health care* (pp. 143–159). Switzerland: Birkhauser Publishing.
- IBM. (2001, October). Autonomic computing, 1–22.
- Ikeda, M. & Mizoguchi, R. (1994). Fits: a framework for its—a computational model of tutoring. *Journal of Artificial Intelligence in Education*, 5(3), 319–48.
- Johnson, C. D. (1972). *Formal aspects of phonological description*. Mouton The Hague.
- Jurafsky, D. & Martin, J. H. (2009). *Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition* (2nd). Upper Saddle River, NJ: Pearson Education.
- Kahn, K. (2004). Toontalk-steps towards ideal computer-based learning environments. In M. Tokoro & L. Steels (Eds.), *A learning zone of one's own: sharing representations and flow in collaborative learning environments* (pp. 253–270). IOS Press.
- Takegawa, J., Kanda, H., Fujioka, E., Itami, M., & Itoh, K. (2000). Diagnostic processing of Japanese for computer-assisted second language learning. In *Proceedings of the 38th annual meeting on association for computational linguistics* (pp. 537–546). Association for Computational Linguistics.
- Kaplan, F. & Oudeyer, P.-Y. (2001). From hardware and software to kernels and envelopes: a concept shift for robotics, developmental psychology, and brain sciences. In *Neuromorphic and brain-based robots* (pp. 217–250). Cambridge: Cambridge University Press.
- Karttunen, L. & Beesley, K. R. (2005). Twenty-five years of finite-state morphology. *Inquiries Into Words, a Festschrift for Kimmo Koskenniemi on his 60th Birthday*, 71–83.
- Kay, P. & Fillmore, C. J. (1999). Grammatical constructions and linguistic generalizations: the what's x doing y? construction. *Language*, 75, 1–33.
- Keuleers, E. & Daelemans, W. (2007). Memory-Based Learning Models of Inflectional Morphology: a Methodological Case Study. *Lingue e Linguaggio*, 6(2).
- Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. In C. Shannon & J. McCarthy (Eds.), *Automata studies* (pp. 3–41). Princeton University Press.
- Knox, J., Bayne, S., Macleod, H., Ross, J., & Sinclair, C. (2012). Mooc pedagogy: the challenges of developing for Coursera. *Association for Learning Technology Newsletter*.
- Koskenniemi, K. (1983a). Two-level model for morphological analysis. In *Ijcai* (Vol. 83, pp. 683–685).
- Koskenniemi, K. (1983b). *Two-level morphology*.

BIBLIOGRAPHY

- Krashen, S. (2013). Rosetta Stone: Does not provide compelling input, research reports at best suggestive, conflicting reports on users' attitudes. *International Journal of Foreign Language Education*, 8(1), 1–3.
- Krashen, S. & Terrell, T. (1983). *The natural approach: language acquisition in the classroom*. New York: Prentice Hall.
- Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: an architecture for general intelligence. *Artificial intelligence*, 33(1), 1–64.
- Lane, J. & Kinser, K. (n.d.). Mooc's and the mcdonaldization of global higher education.
- Lave, J. & Wenger, E. (1991). *Situated learning: legitimate peripheral participation*. Cambridge university press.
- Lesgold, A., Lajoie, S., & Bunzo, M. (n.d.). Eggen. g. 1992. sherlock: a coached practice environment for an electronics troubleshooting job. *Computer Assisted Instruction and Intelligent Tutoring Systems*, 201–238.
- Levin, L. S. & Evans, D. A. (1995). Alice-chan: a case study in icall theory and practice. *Intelligent language tutors: Theory shaping technology*, 77–99.
- Levin, L. S., Evans, D. A., & Gates, D. M. (1991). The alice system: a workbench for learning and using language. *CALICO Journal*, 9(1), 27–56.
- Levy, M. (1997). *Call: context and conceptualisation*. Oxford: Oxford University Press.
- L'Haire, S. & Vandeventer Faltin, A. (2003). Error diagnosis in the freetext project. *Calico Journal*, 20(3), 481–495.
- Lieberman, D. A. (2000). *Learning: behavior and cognition*. Wadsworth Belmont.
- Liebscher, G. D.-O. J. (2003). Conversation repair as a role-defining mechanism in classroom interaction. *The Modern Language Journal*, 87(3), 375–390.
- Liu, T.-C. et al., Peng, H. et al., Wu, W.-H., et al., Lin, M.-S., et al. (2009). The effects of mobile natural-science learning based on the 5e learning cycle: a case study. *Educational Technology & Society*, 12(4), 344–358.
- Loetzsch, M., Wellens, P., De Beule, J., Bleys, J., & van Trijp, R. (2008). *The Babel2 manual* (tech. rep. No. AI-Memo 01-08). AI-Lab VUB. Brussels.
- Lynch, C., Ashley, K., Aleven, V., & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In *Proceedings of the workshop on intelligent tutoring systems for ill-defined domains at the 8th international conference on intelligent tutoring systems* (pp. 1–10).
- Lyster, R. & Ranta, L. (1997, March). Corrective Feedback and Learner Uptake. *Studies in Second Language Acquisition*, 19(01), 37–66.
- MacWhinney, B. (1991). *The CHILDES project: tools for analyzing talk*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Maes, P. (1988). Issues in computational reflection. In P. Maes & D. Nardi (Eds.), *Meta-level architectures and reflection* (pp. 21–35). Amsterdam: Elsevier.
- Maes, P. & Nardi, D. (Eds.). (1988). *Meta-level architectures and reflection*. New York, NY, USA: Elsevier Science Inc.
- Mateo, F. (1998). *Les verbes espagnols*. Hatier.

- Matthews, C. (1992). Going ai: foundations of icall. *Computer Assisted Language Learning*, 5(1-2), 13–31.
- Matthews, C. (1993). Grammar frameworks in intelligent call. *CALICO Journal*, 11(1), 5–27.
- McClelland, J. L. & Patterson, K. (2002). 'Words or Rules' cannot exploit the regularity in exceptions. *Trends in Cognitive Science*, 6(11), 1–2.
- Melissa, H. V., Maisano, R., Alderks, C., & Martin, J. (1993). Parsers in tutors: what are they good for? *CALICO Journal*, 11(1), 28–46.
- Mengelle, T. & Frasson, C. (1996). A multi-agent architecture for an its with multiple strategies. In *Proc. conf. calice'96, Incs 1108* (pp. 96–104).
- Menzel, W. (1988). Error diagnosing and selection in a training system: for second language learning. In *Proceedings of the 12th conference on computational linguistics-volume 2* (pp. 414–419). Association for Computational Linguistics.
- Menzel, W. (1990). Anticipation-free diagnosis of structural faults. In *Proceedings of the 13th conference on computational linguistics-volume 3* (pp. 422–424). Association for Computational Linguistics.
- Menzel, W. (1992). *Modellbasierte fehlerdiagnose in sprachlehrsystemen*. Niemeyer Tübingen.
- Menzel, W. (2006). Constraint-based modeling and ambiguity. *International Journal of Artificial Intelligence in Education*, 16(1), 29–63.
- Menzel, W. & Schröder, I. (1998). Constraint-based diagnosis for intelligent language tutoring systems. In *In proceedings of the it&knows conference at the ifip'98 congress, wien/budapest*.
- Menzel, W. & Schröder, I. (1999). Error diagnosis for language learning systems. *ReCALL*, 11, 20–30.
- Meurers, D. (2012). Natural language processing and language learning. *The Encyclopedia of Applied Linguistics*.
- Micelli, V. (2012). Field topology and information structure: a case study for German constituent order. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- Michaelis, L. A. (2009). Sign-based construction grammar. In B. Heine & H. Narrog (Eds.), *The oxford handbook of linguistic analysis* (pp. 155–176). Oxford: Oxford University Press.
- Michaelis, L. A. & Lambrecht, K. (1996). Toward a construction-based model of language function: the case of nominal extraposition. *Language*, 72, 215–247.
- Michaud, L. N. & McCoy, K. F. (2001). Error profiling: toward a model of english acquisition for deaf learners. In *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 394–401). Association for Computational Linguistics.
- Michaud, L. N. & McCoy, K. F. (2006, January). Capturing the Evolution of Grammatical Knowledge in a CALL System for Deaf Learners of English. *International Journal of Artificial Intelligence in Education*, 16(1).

BIBLIOGRAPHY

- Mislevy, R. J. & Yin, C. (2009). If Language Is a Complex Adaptive System, What Is Language Assessment? *Language Learning*, 59(s1), 249–267.
- Mitchell, R., Dominguez, L., Maria, A., Myles, F., & Marsden, E. (2008). A new database for Spanish second language acquisition research. *EUROSLA Yearbook*, 8(1), 287–304.
- Mitrovic, A. & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10(3-4), 238–256.
- Müller, S. (1996). The babel-system – an HPSG prolog implementation. In *Proceedings of the fourth international conference on the practical application of prolog* (pp. 263–277). London.
- Murphy Paul, A. (2012, September). How computerized tutors are leaning to teach humans. New York Times.
- Murphy, M., Krüger, A., & Grieszl, A. (1998). Recall-providing an individualized call environment. *Language Teaching and Language Technology, Lisse: Swets & Zeitlinger*, 62–73.
- Murphy, M. & McTear, M. (1997). Learner modelling for intelligent call. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: proceedings of the sixth international conference* (pp. 301–312). Wien, New York: Springer Verlag.
- Nerbonne, J. A. (2003). Computer-assisted language learning and natural language processing. In R. Mitkov (Ed.), *The oxford handbook of computational linguistics* (pp. 670–698). Oxford: Oxford University Press.
- Ng, A. & Koller, D. (2012). The online revolution: education at scale.
- Nkambou, R., Bourdeau, J., & V., P. (2010). Building Intelligent Tutoring Systems: An Overview. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems* (Chap. 18, pp. 361–375). Studies in Computational Intelligence. Berlin: Springer.
- Nkambou, R. & Kabanza, F. (2001). Designing intelligent tutoring systems: a multiagent planning approach. *ACM SIGCUE Outlook*, 27(2), 46–60.
- Ohlsson, S. (1994). Constraint-based student modelling. In J. E. Greer & G. I. McColla (Eds.), *Student modelling: the key to individualized knowledge-based instruction* (pp. 167–189). Berlin: Springer.
- O'malley, J. M. & Chamot, A. U. (1990). *Learning strategies in second language acquisition*. Cambridge University Press.
- CTAT Basics. (2013, September). Retrieved from http://ctat.pact.cs.cmu.edu/docs/ctat_2_0/ch01s03.html
- Ozer, O. (2004, October). Constructivism in piaget and vygotsky. *Fountain*, (48), online.
- Panova, I. & Lyster, R. (2002). Patterns of corrective feedback and uptake in an adult ESL classroom. *Tesol Quarterly*, 36(4), 573–595.
- Papert, S. (1980). *Mindstorms: children, computers and powerful ideas*. New York: Basic Books.

- Papert, S. (1987). Microworlds: transforming education. In R. Lawler & M. Yazdani (Eds.), *Artificial intelligence and education* (Chap. 4, Vol. 1, pp. 79–94). Norwood, NJ: Ablex.
- Pappano, L. (2012). The year of the mooc. *The New York Times*.
- Pardos, Z., Gowda, S., Baker, R., & Heffernan, N. (2012). The sum is greater than the parts: ensembling models of student knowledge in educational software. *ACM SIGKDD Explorations Newsletter*, 13(2), 37–44.
- Park Woolf, B. (2008). *Building intelligent interactive tutors*. Burlington, MA: Morgan Kaufmann.
- Pauw, S. & Hilferty, J. (2012). The emergence of quantifiers. In L. Steels (Ed.), *Experiments in Cultural Language Evolution* (pp. 277–304). John Benjamins.
- Payne, S. J. & Squibb, H. R. (1990). Algebra mal-rules and cognitive accounts of error. *Cognitive Science*, 14(3), 445–481.
- Philippa, M., Debrabandere, F., Quak, A., Schoonheim, T., & van der Sijs, N. (2009). *Etymologisch woordenboek van het nederlands*. Amsterdam: Amsterdam University Press.
- Pollard, C. & Sag, I. A. (1994). *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.
- Polson, M. & Richardson, J. (2013). *Foundations of intelligent tutoring systems*. Interacting with Computers Series. Taylor & Francis.
- Postma, A. & Kolk, H. (1993). The covert repair hypothesis: prearticulatory repair processes in normal and stuttered disfluencies. *Journal of Speech, Language and Hearing Research*, 36(3), 472.
- Razek, M. A., Frasson, C., & Kaltenbach, M. (2002). Toward more cooperative intelligent distance learning environments. *Software Agents Cooperation Human Activity*.
- Razzaq, L. & Heffernan, N. (2009). To tutor or not to tutor: that is the question. In *Proceedings of the conference on artificial intelligence in education* (pp. 457–464).
- Razzaq, L. & Heffernan, N. (2010). Open content authoring tools. *Advances in Intelligent Tutoring Systems*, 407–420.
- Razzaq, L., Patvarczki, J., Almeida, S., Vartak, M., Feng, M., Heffernan, N., & Koedinger, K. (2009). The assistment builder: supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on*, 2(2), 157–166.
- Rello, L. & Basterrechea, E. (2011). Onoma: A Linguistically Motivated Conjugation System for Spanish Verbs, New York: Springer, 6608, 227–138. CICLing 2011: The 12th International Conference on Intelligent Text Processing and Computational Linguistics, Lecture Notes in Computer Science.
- Reuer, V. (2003). Error recognition and feedback with lexical functional grammar. *CALICO journal*, 20(3), 497–512.
- Rinaldi, C. (2003). The joys of preschool learning. In M. Tokoro & L. Steels (Eds.), *The future of learning* (pp. 57–71). IOS Press.

BIBLIOGRAPHY

- Rosetta Stone. (2007, December). *Revolutionary Technology Strengthens Military Language Training*. Retrieved from <http://pr.rosettastone.com/phoenix.zhtml?c=228009&p=irol-newsArticle&ID=1273960>
- Rumelhart, D. E. & McClelland, J. L. (1986, January). On learning the past tenses of english verbs. *Parallel distributed processing*.
- Ryan, R. M. & Deci, E. L. (2000). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemporary educational psychology*, 25(1), 54–67.
- Rypa, M. & Feuerman, K. (1995). Calle: an exploratory environment for foreign language learning. *Intelligent language tutors: theory shaping technology*, 55–76.
- Sag, I. A. (2012). Sign-based construction grammar: An informal synopsis. In H. C. Boas & I. A. Sag (Eds.), *Sign-based construction grammar* (pp. 61–188). Sign-Based Construction Grammar, Stanford: CSLI Publications.
- Salah, D. & Zeid, A. (2009). Plits: a pattern language for intelligent tutoring systems. In *Europlop*.
- Sandberg, J., Maris, M., & de Geus, K. (2011, August). Mobile English learning: An evidence-based study with fifth graders. *Computers & Education*, 57(1), 1334–1347.
- Schneider, N. (2010). Computational Cognitive Morphosemantics: Modeling Morphological Compositionality in Hebrew Verbs with Embodied Construction Grammar. *36th Annual Meeting of the Berkeley Linguistic Society (BLS)*. Berkeley, CA.
- Schulze, M. (1999). From the developer to the learner: describing grammar–learning grammar. *ReCALL*, 11(01), 117–124.
- Schulze, M. (2001). *Textana: grammar and grammar checking in parser-based call* (Doctoral dissertation, UMIST, Manchester).
- Schulze, M. & Hamel, M.-J. (1998). Use and re-use of syntactic parsers in call. towards diagnosing learner errors. *Proceedings of WorldCALL*, 203–204.
- Schulze, M. & Penner, N. (2008). Construction grammar in icall. *Computer Assisted Language Learning*, 21(5), 427–440.
- Self, J. A. (1974). Student models in computer aided instruction. *International journal of man-machine studies*, 6, 261–276.
- Self, J. A. (1977). Concept teaching. *Artificial Intelligence*, 9, 197–221.
- Self, J. A. (1985). A perspective on intelligent computer-aided instruction. *Journal of Computer Assisted Learning*, 1, 159–166.
- Self, J. A. (1988). Bypassing the intractable problem of student modeling. In C. Frasson & G. Gauthier (Eds.), *Intelligent tutoring systems: at the crossroads of artificial intelligence and education*. Norwood, NJ: Ablex.
- Self, J. A. (1990). Theoretical foundations for intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 1(4), 3–14.
- Shang, Y., Shi, H., & Chen, S.-S. (2001). An intelligent distributed environment for active learning. *Journal on Educational Resources in Computing (JERIC)*, 1(2es), 4.

- Sibley, E. H., Pollock, J. J., & Zamora, A. (1984). Automatic Spelling Correction in Scientific and Scholarly Text. *Communications of the ACM*, 27(4).
- Siemens, G. (n.d.). The best learning of my life.
- Sison, R. & Shimura, M. (1998). Student modeling and machine learning. *International Journal of Artificial Intelligence in Education (IJAIED)*, 9, 128–158.
- Sivell, J. (2004). Tools for embodied teaching: célestin freinet and the learner-centered classroom. In M. Tokoro & L. Steels (Eds.), *A learning zone of one's own* (pp. 155–170). IOS Press.
- Skinner, B. F. (1961). *Cumulative record*. New York: Appleton Century Crofts.
- Sleeman, D. [D.]. (1987). Pixie: a shell for developing intelligent tutoring system. In R. L. M. Yazdani (Ed.), *Ai education* (Vol. 1, pp. 239–265). London: Chapman Hall.
- Sleeman, D. [Derek]. (1982). Inferring (mal) rules from pupils' protocols. In *Selected and updated papers from the proceedings of the 1982 european conference on progress in artificial intelligence* (pp. 30–39). John Wiley & Sons, Inc.
- Smith, B. C. (1982). *Procedural reflection in programming languages* (Doctoral dissertation, Massachusetts Institute of Technology).
- Snyder, B. & Barzilay, R. (2008). Unsupervised multilingual learning for morphological segmentation. In *Proceedings of acl-08: hlt*.
- Socher, R., Lin, C. C., Ng, A., & Manning, C. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 129–136).
- Spiegler, S., Golenia, B., & Flach, P. (2009). Promodes: A probabilistic generative model for word decomposition. *Working Notes for the CLEF 2009*
- Spranger, M. & Loetzsch, M. (2011). Syntactic indeterminacy and semantic ambiguity: a case study for German spatial phrases. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 265–298). Amsterdam: John Benjamins.
- Steels, L. (2003). Language re-entrance and the 'inner voice'. *Journal of Consciousness Studies*, 10(4-5), 173–185.
- Steels, L. (2004). The architecture of flow. In M. Tokoro & L. Steels (Eds.), *A learning zone of one's own* (pp. 137–149). IOS Press.
- Steels, L. (Ed.). (2011). *Design patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.
- Steels, L. (Ed.). (2012a). *Computational issues in Fluid Construction Grammar*. Lecture Notes in Artificial Intelligence. Berlin: Springer.
- Steels, L. (2012b). Design methods for Fluid Construction Grammar. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- Steels, L., De Beule, J., & Wellens, P. (2012). Fluid construction grammar on real robots. In L. Steels & M. Hild (Eds.), *Language Grounding in Robots* (pp. 195–213). Berlin, Heidelberg: Springer.

BIBLIOGRAPHY

- Steels, L. & Loetzsch, M. (2010). Babel: a tool for running experiments on the evolution of language. In S. Nolfi & M. Mirolli (Eds.), *Evolution of communication and language in embodied agents* (pp. 307–313). Berlin: Springer Verlag.
- Steels, L. & van Trijp, R. (2011). How to make construction grammar fluid and robust. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 301–330). Amsterdam: John Benjamins.
- Steels, L. & Wellens, P. (2006). How grammar emerges to dampen combinatorial search in parsing. In P. Vogt, Y. Sugita, E. Tuci, & C. Nehaniv (Eds.), *Symbol grounding and beyond. proceedings of the third eelc* (pp. 76–88). LNAI 4211. Berlin: Springer-Verlag.
- Stockwell, G. (2008). Investigating learner preparedness for and usage patterns of mobile learning. *ReCALL*, 20(3), 253–270.
- Sun, S., Joy, M., & Griffiths, N. (2007). The use of learning objects and learning styles in a multi-agent education system. *Journal of Interactive Learning Research*, 18(3), 381–398.
- Tomasello, M. (2003). *Constructing a language. a usage based theory of language acquisition*. Harvard University Press.
- Turner, T., Macasek, M., Nuzzo-Jones, G., Heffernan, N., & Koedinger, K. (2005). The assistent builder: a rapid development tool for its. In *Proceedings of the 12th annual conference on artificial intelligence in education* (pp. 929–931).
- Uhr, L. (1969). Teaching machine programs that generate problems as a function of interaction with students. In *Proceedings of the 24th national conference*, (pp. 125–134).
- Van den Bosch, A. & Daelemans, W. (1999). Memory-based morphological analysis. In *Proceedings of the 37th annual meeting of the association for computational linguistics* (pp. 285–292). Morristown, NJ, USA: Association for Computational Linguistics.
- van Trijp, R. (2011). Feature matrices and agreement: a case study for German case. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 205–235). Amsterdam: John Benjamins.
- van Trijp, R. (2012). A reflective architecture for language processing and learning. In L. Steels (Ed.), *Computational Issues in Fluid Construction Grammar*. Berlin: Springer Verlag.
- van Trijp, R. (2013, January). A comparison between Fluid Construction Grammar and Sign-Based Construction Grammar. *Constructions and Frames*, 5(1), 88–116.
- van Trijp, R. (2010). Strategy competition in the evolution of pronouns: a case-study of spanish leísmo, láismo and loísmo. In A. Smith, M. Schouwstra, B. de Boer, & K. Smith (Eds.), *The evolution of language (evolang 8)* (pp. 336–343). Singapore: World Scientific.
- Vanden Bulke, N. D. D. M. P. (2005). *El uso de los tiempos del pasado*. Wolters Plantyn.
- Vandeventer Faltin, A. (2003). Natural language processing tools for computer assisted language learning. *Linguistik online*, 17(5), 03.

- Vandeventer, A. (2000). Research on nlp-based call at the university of geneva. *TELL & CALL*, 2/2000, 10–13.
- Vandeventer, A. (2001). Creating a grammar checker for call by constraint relaxation: a feasibility study. *ReCALL*, 13(01), 110–120.
- Vandeventer, A. & Hamel, M.-J. (2000). Reusing a syntactic generator for call purposes. *ReCALL*, 12(01), 79–91.
- Vandewaetere, M., Desmet, P., & Clarebout, G. (2011, January). Computers in Human Behavior. *Computers in Human Behavior*, 27(1), 118–130.
- VanLehn, K. (1988). Student Modeling. In M. Polson & J. Richardson (Eds.), *Foundations of intelligent tutoring systems* (pp. 55–78). Hillsdale, NJ: Erlbaum.
- VanLehn, K. (2007). Intelligent tutoring systems for continuous, embedded assessment. In C. Dwyer (Ed.), *The future of assessment: shaping teaching and learning*. Lawrence Erlbaum Associates.
- Vassileva, J., Mccalla, G., & Greer, J. (2003). Multi-agent multi-user modeling in i-help. *User Modeling and User-Adapted Interaction*, 13(1), 179–210.
- Vesselinov, R. & Grego, J. (2012, December). *Duolingo Effectiveness Study*.
- Vicari, R. & Gluz, J. (2007). An intelligent tutoring system (its) view on aose. *International Journal of Agent-Oriented Software Engineering*, 1(3), 295–333.
- Vivet, M. (1988). Knowledge-based tutors. towards the design of a shell. *International Journal of Educational Research*, 12(8), 839–850.
- von Ahn, L. (2013a). Augmented intelligence: the web and human intelligence. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987).
- von Ahn, L. (2013b). Duolingo: learn a language for free while helping to translate the web. In *Proceedings of the 2013 international conference on intelligent user interfaces* (pp. 1–2). ACM.
- Vygotsky, L. (1978). *Mind in society*. Cambridge, MA: Harvard University Press.
- Walonoski, J. & Heffernan, N. (2006). Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In *Intelligent tutoring systems* (pp. 382–391). Springer.
- Waterman, M., Matlin, K., & P.A., D. (1993). Using cases for teaching and learning in the life sciences: an example from cell biology. *Coalition for education in the life sciences*, Woods Hole, MA.
- Wauters, K., Desmet, P., & Van den Noortgate, W. (2010, September). Adaptive item-based learning environments based on the item response theory: possibilities and challenges. *Journal of Computer Assisted Learning*, 26(6), 549–562.
- Weinberg, A., Garman, J., Martin, J., & Merlo, P. (1995). A principle-based parser for foreign language tutoring in german and arabic. *Intelligent language tutors: theory shaping technology*, 23–44.
- Weinstein, C. E. & Mayer, R. E. (1986). The teaching of learning strategies. *Handbook of research on teaching*, 3, 315–327.

BIBLIOGRAPHY

- Weischedel, R. M., Voge, W. M., & James, M. (1978). An artificial intelligence approach to language instruction. *Artificial Intelligence*, 10(3), 225–240.
- Wellens, P. (2011). Organizing constructions in networks. In *Design patterns in Fluid Construction Grammar* (pp. 181–201). Amsterdam: John Benjamins.
- Wellens, P. (2012). *Adaptive strategies in the emergence of lexical systems* (Doctoral dissertation, Vrije Universiteit Brussel).
- Wellens, P. & De Beule, J. (2010). Priming through constructional dependencies: a case study in Fluid Construction Grammar. In A. Smith, M. Schouwstra, B. de Boer, & K. Smith (Eds.), *The evolution of language (evolang8)* (pp. 344–351). Singapore: World Scientific.
- Whitley, M. (2002). *Spanish/english contrasts: a course in spanish linguistics*. Georgetown University Press.
- Wood Bowden, H., Gelfand, M. P., Sanz, C., & Ullman, M. T. (2010). Verbal Inflectional Morphology in L1 and L2 Spanish. *Language Learning*, 60(1), 44–87.
- Woolf, B., Arroyo, I., Cooper, D., Burleson, W., & Muldner, K. (2010). Affective tutors: automatic detection of and response to student emotion. *Advances in Intelligent Tutoring Systems*, 207–227.
- Yannakoudakis, E. & Fawthrop, D. (1983). The rules of spelling errors. *Information Processing & Management*, 19(2), 87–99.
- Yu, C., Smith, L., & Pereira, A. (2008). Grounding word learning in multimodal sensorimotor interaction. In B. Love, K. McRae, & V. Sloutsky (Eds.), *Proceedings of the 30th annual conference of the cognitive science society*. Cognitive Science Society. Austin, TX.
- Yu, C. & Smith, L. B. (2012). Embodied attention and word learning by toddlers. *Cognition*.
- Zeid, A. & Salah, D. (2010). Using pattern languages to design intelligent tutoring systems: a case study. *IJCSNS*, 10(9), 164.

Publications

Because this dissertation is the resulting work of only a part of my total time as a PhD student, I hereby summarise my main publications since the start of my PhD degree (October 2009). The publications are presented according to four main research topics. Nevertheless, the research that is not related to this dissertation has helped me to understand and further develop the tools and frameworks that were used to carry out the work described in this dissertation.

- **Fluid Construction Grammar** The Fluid Construction Grammar (FCG) formalism has been jointly developed by researchers at the VUB AI Lab and the Sony Computer Science Lab in Paris. By developing case studies in FCG that treated complex grammatical structures in particular languages, I learned about the power of the system and helped to expand it where necessary. Moreover, I was responsible for adding an additional layer of flexibility to the system so that unexpected use of particular constructions could be diagnosed and repaired during processing.

Wellens, P., R. van Trijp, **Beuls, K.**, Steels, L. (2013). Fluid Construction Grammar for Historical and Evolutionary Linguistics. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (p. 127–132). Sofia, Bulgaria, August 4–9 2013.

Beuls, K. (2013). Inflectional Patterns as Constructions: Spanish verb morphology in Fluid Construction Grammar. *Constructions and Frames*, 4:2. Amsterdam: John Benjamins

Beuls, K., van Trijp, R., Wellens, P. (2012). Diagnostics and Repairs in Fluid Construction Grammar. In L. Steels M. Hild (Eds.), *Language Grounding in Robots*. Berlin: Springer.

Beuls, K. (2012). Construction Sets and Unmarked Forms: A Case Study

for Hungarian Verbal Agreement. In L. Steels (Ed.), *Design Patterns in Fluid Construction Grammar* (p. 237-264). Amsterdam: John Benjamins.

van Trijp, R., Steels, L., **Beuls, K.**, Wellens, P. (2012). Fluid Construction Grammar: The New Kid on the Block. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon.

Beuls, K. (2012). Spanish Modals in Fluid Construction Grammar. In L. Steels (Ed.), *Computational Issues in Fluid Construction Grammar*. Berlin: Springer.

Beuls, K., Wellens, P. (2010). Linking Constructions and Categories: A Case Study for Hungarian Object Agreement. In *Book of Abstracts of the Sixth International Conference on Construction Grammar* (p. 44–45).

- **Error diagnosis** I published the first application of Fluid Construction Grammar for the diagnosis and correction of ungrammatical utterances. The ungrammaticality comes in these articles from the non-proficient Spanish verb conjugation by learners whose utterances are recorded in the Spanish Learner Language Oral Corpora (SPLLOC).

Beuls, K. (2012). Grammatical error diagnosis in Fluid Construction Grammar: a case study in L2 Spanish verb morphology. *Computer Assisted Language Learning*, 1–15. doi:10.1080/09588221.2012.724426

Beuls, K. (2012). Construction Grammar Towards an L2 grammar corrector for Spanish: Tense, aspect and mood in Fluid Construction Grammar. EUROCALL 2012, Gothenburg. *This paper was not included in the proceedings due to my absence at the conference.*

- **Language tutoring and language games** Already since the start of my PhD degree, I have been focusing on language acquisition and the role of situated learning herein. My first conference paper (Benelearn 2010) discussed learning operators needed to simulate the acquisition of Hungarian poly-personal agreement. Other computational simulations focus on the acquisition of Russian aspect a first prototype of a tutoring system for the domain of colour language (together with Joris Bleys).

Beuls, K. (2013). A constructionist approach to student modeling: tracing a student's constructions through an agent-based tutoring architecture. *EU-ROCALL Conference proceedings: Learning from the Past, Looking to the Future*. Evora, Portugal, 11–14 September, 2013.

Gerasymova, K., Spranger, M., **Beuls, K.** (2012). A Language Strategy for Aspect: Encoding Aktionsarten through Morphology. In L. Steels (Ed.), *Experiments in Cultural Language Evolution*. Amsterdam: John Benjamins.

Beuls, K., Bleys, J. (2011). Game-based Language Tutoring: A Case Study for Colour. In *Proceedings of the IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)* (p. 1–5).

Beuls, K., Gerasymova, K., van Trijp, R. (2010). Situated Learning through the Use of Language Games. In *Proceedings of the 19th Annual Machine Learning Conference of Belgium and The Netherlands* (p. 1–6).

Beuls, K., Bleys, J., Micelli, V. (2009, October). *Colour Language Tutoring*. Poster session presented at the Sony Computer Science Lab Open House, Paris.

- **Agreement** One line of research that was developed over the last years concerns the emergence and evolution of grammatical systems such as wide-spread linguistic phenomenon of grammatical agreement. A language disposes of an agreement system when features of one linguistic unit are expressed on a different unit, within the same utterance or across different utterances. The articles in this section investigate the function of agreement and the strategies that exist to express agreement across different languages.

Steels, L., **Beuls, K.** (in press). Explaining the origins of complexity in language. In S. Mufwene (Ed.), *Complexity in Language*. Cambridge: Cambridge University Press.

Beuls, K., Steels, L. (2013). Agent-based models of strategies for the emergence and evolution of grammatical agreement. *PLOS ONE*.

Beuls, K., Steels, L., Höfer, S. (2012). The Emergence of Internal Agreement Systems. In L. Steels (Ed.), *Experiments in Cultural Language Evolution*. Amsterdam: John Benjamins.

Beuls, K. (2011). Disambiguating Stickers. Grammatical Agreement as a

Design Pattern. In *Proceedings of the Twenty-third Benelux Conference on Artificial Intelligence* (p. 1–8).

Beuls, K., Höfer, S. (2011). Simulating the Emergence of Grammatical Agreement in Multi-Agent Language Games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (p. 61–66).

Beuls, K. (2011, June). *The Emergence of Nominal Constituency*. Poster session presented at the LOT Summer School, Leuven.

A complete list of my research activities including oral presentations, posters, workshop and conference attendances and external research stays can be found online at ai.vub.ac.be/members/katrien/activities.

Glossary

Alignment In a language game alignment happens after the interaction when speaker and hearer try to align their construction inventories. In a tutoring game, alignment takes place on a different level, namely between the student agent and the student, after the tutor feedback has been produced. Alignment is a vital process because it brings the student model closer to the real student so that the student's actions can be predicted better.

Construction inventory The construction inventory is a flat list of constructions, which are conventionalized form-meaning mappings in a language. Constructions all share the same basic data structure but their functions might differ.

Game See tutoring game.

Grammar engine The grammar engine is the motor to configure the construction inventory and process utterances. Competent language agents optimize their grammar engine so that production and parsing processes run as smooth as they can.

Language agent The language agent is a model of a competent language user of the language system. This agent can produce and parse utterances that correspond to his grammar and he will notify ungrammatical parsing solutions when he is interacting with a student agent. The language agent is capable of teaching the grammar of the target language system to this learner.

Learning problem A learning problem is the core of a learning strategy, detectable by a diagnostic and resolvable by a repair. It contains the following fields: a problem type, problem instances, typical features and a learning problem priority score. Learning problems are also used by the student profile and the tutoring strategies to keep track of the real student's knowledge of the target language system.

Language game A language game can be considered as a microworld that operationalizes everything needed for modeling a routinized, communicative interaction: a situated context, two (or more) interlocutors, a communicative purpose, and so on. In a single game, the interlocutors go through multiple steps: conceptualization/interpretation, linguistic processing and learning and alignment.

Language system A language system is a particular grammatical subsystem in a language, such as its system for agreement marking, argument structure, tense-aspect-mood system, etc. Language systems try to maximise expressive power and communicative success but they must also minimise cognitive effort, balancing the cost of maintaining an additional grammatical system like determiners versus the cost of additional processing and lengthier utterances.

Learning strategy A learning strategy is a pair of a diagnostic and a repair that is used to tackle a particular learning problem. By default there is only one repair for every diagnostic but sometimes we find sets of repairs for a single diagnostic. They can then either be in parallel competition, with their scores defining the order of application, or they can be carried out in a subsequent fashion. Learning strategies are always specific to one language or language system.

Session See tutoring session.

Student Sometimes called real student or learner, this term is used to refer to the actual user that interacts with the tutoring system. The utterances of the student can at any moment be compared with what the student agent would have said or understood.

Student agent The student agent has the same architecture as the language agent and possesses a construction inventory, a grammar engine to process utterances and meta-level operators. Unlike a language agent, the latter are not used as flexibility strategies but serve as language learning strategies instead, that can also modify the construction inventory and the grammar engine. The student agent is thought to be a model of the real student in that it can be used to predict his utterances.

Student model A student model should be as close as possible to the real student. The student model presented in this dissertation consists of an operational student agent that can be run to simulate the real student's behaviour and a more static student profile that contains user information, measures and log data of the student.

Student profile The student profile is a subpart of the student model and contains four elements: general user information, log data of the games that have been

played, a priority list with learning problems to be tackled and measures that reflect the student's skill level. The student profile is heavily consulted by the tutoring strategies when selecting a new situation and providing feedback.

Tutor agent The tutor agent is the general term to refer to the combination of language agent, student model and tutoring strategies. The tutor agent contains thus two "subagents", namely the language agent and the student agent, that can also play tutoring games even without the presence of a real student. Tutoring strategies and the student profile are uniquely used with a human learner in the loop.

Tutoring game A tutoring game is a type of language game in which one of the two interacting agents is a competent language agent and the other one is an incompetent language user, either a student agent or a human student. Similar to a language game, game participants share a situation (not yet conceptualized) and a joint goal, such as expressing a certain conceptualization of the situation or pointing to the situation's topic.

Tutoring session A tutoring session is a series of tutoring games played by the same student without any interruptions. When a student logs out, the session ends and a new session can be started the next time he enters the system. He is recognized through his user name, which is kept in his student profile.

Tutoring strategy A tutoring strategy is a means to tackle a particular learning problem in a student's acquisition of the target language that is diagnosed by the tutor agent. It influences the learning path of the student by selecting new learning situations and providing feedback at the end of every language game.

Index

- ==, 20
- ==0, 20
- ==1, 20
- ==p, 20
- absolute tense, 34
- assimilation, 33
- automatic grammar creation, 91
- base grammar, 96
- computational introspection, 55
- computational reflection, 54, 55
- construction, 14, 15
- construction application, 25
- Construction Grammar, 13
- construction inventory, 13
- construction network, 22
- construction organization, 20
- construction poles, 15
- construction sets, 21
- construction templates, 23, 83
- construction types, 35
- constructional dependency, 22
- constructivism, 80
- convergence, 112
- conversational repair strategies, 87
- coupled feature structure, 15
- decision tree, 91
- diagnostics, 57–60
- diagnostics and repairs, 99
- diphthongization, 33
- Embodied Construction Grammar, 12
- FCG, *see* Fluid Construction Grammar
- feature matrix, 35
- Finite-state morphology, 51
- finite-state transducer, 51
- flexibility strategy, 57
- Fluid Construction Grammar, 12–13
- footprints, 19
- goal test, 26
- grammar engine, 25
- hashed construction set, 97
- Head-driven Phrase Structure Grammar, 12
- ICALL, 68
- initial feature structure, 26
- irregular verbs, 45
- J-operator, 17, 25
- J-unit, 17
- language agent, 3
- learning strategies, 99
- learning strategy, 99
- lexical constructions, 35
- match, 25
- merge, 25
- meta-level architecture, 58
- morphological constructions, 39
- Onoma, 91
- Open learner models, 132
- parsing, 25
- phonological constructions, 40, 48
- phrasal constructions, 38
- primary verbs, 92
- problem, 57

problem-driven learning, 99
processing constructions, 25
production, 25
programmed instruction, 151

re-entrance, 65
reflective architecture, 56
reflective language processing, 56
regular verbs, 42
relative tense, 34
render-robust, 47
rendering, 47
repairs, 57, 60–61
robustness, 56
Rosetta Stone, 100

search process, 26
semi-regular, 92
semi-regular verbs, 44
Sign-based Construction Grammar, 12
SOAR, 55
special operators, 20
SPLLOC, 73
student agent, 5

tag operator, 18
templates, *see* construction templates
top unit, 26
transient linguistic structure, 16
tutor agent, 6
tutoring strategies, 6
Two-level Morphology, 51

unmarked cases, 22

variable, 15, 16
verb ending, 31
verb types, 33