Proof by Resolution

Geraint A. Wiggins
Professor of Computational Creativity
Department of Computer Science
Vrije Universiteit Brussel

Objectives

Queen Mary
University of London

- Learn about general-purpose theorem proving in predicate calculus

The Problem

- Given
  - a knowledge base, KB (a set of sentences, S, and an interpretation, I)
- Prove
  - a sentence, α (under the same interpretation, I)
- Formally
  - Show that KB $\models$ α
    - KB *entails* α
    - α *follows from* KB

The Toolbox

Queen Mary University of London

- Modus ponens
  - Given $\{ p \rightarrow q, p \} \subset KB$, is q true?
  - Yes: $\{ p \rightarrow q, p \} \models \{ q \}$
- Modus Tollens
  - Given $\{ p \rightarrow q, \neg q \} \subset KB$, is p true?
  - No: $\{ p \rightarrow q, \neg q \} \models \{ \neg p \}$
- We can form arbitrarily long "chains" of inference to prove a sentence
- We can reason
  - forwards from what we know to what we want to prove
  - backwards from what we want to prove to what we know
    - backwards is generally more efficient: no search branches leading off-topic

Theorem Proving

- A theorem proving process involves choosing and applying such rules until the desired sentence is shown to be entailed
- It's called a *proof* because the rules used are known, a priori, to be *sound* (i.e., correct)
- However, choice of rule is hard, because you can't know that a particular rule chosen from a range will turn out to be the right one, in a long proof
  - e.g., Modus Ponens is incomplete
  - therefore, each time we use it, we also have to consider other possibilities
  - therefore, each time we use it, we create a set of alternative choices

Modus Ponens is incomplete

Queen Mary University of London

- Consider these rules:
  - If it is raining (R), I will carry an umbrella (U)
  - If it is not raining (¬R), I will carry an umbrella (U)
- It is easy to conclude (as a human) that I always carry an umbrella
  - $\{ R \rightarrow U, \neg R \rightarrow U \} \models \{ U \}$
  - but this isn't provable using modus ponens alone
    - we'd need the law of excluded middle: $R \vee \neg R$
- However, there is a more general rule, that is *complete*

Resolution

- Unit resolution $\quad (P \vee Q) \wedge \neg Q \rightarrow P \qquad \dfrac{P \vee Q \qquad \neg Q}{P}$
  - $\{ P \vee Q, \neg Q \} \models \{ P \}$
- Generalised resolution $\quad (P \vee Q) \wedge (R \vee \neg Q) \rightarrow (P \vee R) \qquad \dfrac{P \vee Q \qquad R \vee \neg Q}{P \vee R}$
  - $\{ P \vee Q, R \vee \neg Q \} \models \{ P \vee R \}$
- Example: Umbrella again

$$\textit{resolution}\ \dfrac{\textit{defn}\ \dfrac{R \rightarrow U}{\neg R \vee U} \qquad \neg\ \textit{elim}\ \dfrac{\textit{defn}\ \dfrac{\neg R \rightarrow U}{\neg\neg R \vee U}}{R \vee U}}{\vee\ \textit{elim}\ \dfrac{U \vee U}{U}}$$

Resolution refutation

Queen Mary
University of London

- This simple rule can be used as follows
  - add the negation of the sentence, S, to be proven to the KB
  - see if this leads to a contradiction
- This idea applies the law of the excluded middle
  - if $\neg S$ is inconsistent with KB, then KB $\models$ S
- This is called *resolution refutation*
  - this is the basis of the "Logic Programming" language, Prolog

Resolution Refutation

- Notation
  - note that the premises are brought in when needed, not all at the top

Premise 2

Premise 1

$$
\dfrac{
  \dfrac{\dfrac{R \rightarrow U}{\neg R \vee U}\;\textit{defn}
  \qquad
  \dfrac{\dfrac{\neg R \rightarrow U}{\neg\neg R \vee U}\;\textit{defn}}{R \vee U}\;\neg\ \textit{elim}}
  {\dfrac{U \vee U \qquad \neg U}{U}\;\textit{resolution}}\;\textit{resolution}
  \qquad \neg U
}{\{\}}\;\textit{resolution}
$$

Negated goal

Empty sentence

Clausal Form

- Resolution is a single, simple, sound, complete rule
  - But we had to do some manipulation first, to get the sentences into a form on which we could use it
- This is *clausal normal form (CNF)*
  - Note that CNF also stands for Conjunctive Normal Form
- Putting FOPC sentences into clausal form is a mechanical procedure that can be done without search

Clausal Normal Form in FOPC

1. Rewrite $\rightarrow$: $a \rightarrow b \Rightarrow \neg a \vee b$

2. Minimise the scope of negations using logical definitions given before
   - $\neg\exists x.A(x) \Rightarrow \forall x. \neg A(x)$
   - $\neg\forall x.A(x) \Rightarrow \exists x.\neg A(x)$
   - $\neg(A \vee B) \Rightarrow \neg A \wedge \neg B$
   - $\neg(A \wedge B) \Rightarrow \neg A \vee \neg B$
   - etc.
   - Note that in this case, these definitions are *unidirectional*, so they are *confluent*
     - no need for searching through alternatives

3. Rewrite remaining double negations: $\neg\neg A \Rightarrow A$

Clausal Normal Form in FOPC

Queen Mary
University of London

4. Standardise variables apart

- rename all quantified variables so that each quantifier is associated with a different name, regardless of its scope

5. *Skolemise* all existential quantifiers

- A Skolem constant is a made-up name for an object that must exist (even though we don't know what it is)
  - $\exists x.P(x) \Rightarrow P(A)$ where A is an arbitrary object in the allowable substitutions of x
  - Use a different arbitrary object for each quantifier

6. Drop all universal quantifiers

- At this point, all variables are universally quantified, because we Skolemised the existentials, so we no longer need to say so explicitly

Clausal Normal Form in FOPC

Queen Mary University of London

7. Convert the sentence into *conjunctive normal form*
   - a sentence in CNF is a conjunction of disjunctions of atomic sentences
     - recall that the resolution rule works on disjunctions
   - do this by rewriting under logical rules
     - de Morgan's laws
     - distributivity of $\wedge$ and $\vee$
8. Split the top-level conjunction up, to make a set of disjunctions
9. Standardise the variables apart again, w.r.t. clauses
   - so that x in a given clause is not the same as x in another clause

- Use the resulting set of clauses as KB in a resolution proof

First Order Term Unification

- As a result of Skolemizing and removing universal quantifiers, we can introduce a new procedure for assigning values to variables
- This is related to $\forall$ instantiation in the standard FOPC
  - effectively, we use the relevant Skolem constants to instantiate the variables
- There is a deterministic algorithm for unification
- The idea is to use literals of a given predicate which contain information about the values of variables (i.e., Skolem constants or functions) to deduce the values of variables in other literals of the same predicate
  - e.g. Father(x, y) and Father(John, Jim) are unified to Father(John, Jim)
  - with a *unifier* (or substitution set) of { John/x, Jim/y }
    - notation: a/x means "a replaces x"

First Order Term Unification

- Notation
  - We sometimes write *Term Unifier* to mean "The result of applying this *unifier* to this *term*"
    - P(x, y) { A/x, B/y } which evaluates to P(A, B)
- Successive application of unifiers
  - We can write *Term Unifier1 Unifier2* to mean "The result of applying these unifiers, one at a time, to Term"
    - P(x, y) { A/x }{ B/y } which evaluates to P(A, B)
- Composition of unifiers
  - We can combine unifiers, so long as there are no contradictory assignments
    - { A/x } { B/y } combine to give { A/x, B/y }
    - { A/x } { B/x } do not combine, because x would have to take 2 different values at once

First Order Term Unification

- To unify two terms (or literals):
    1. if either is a variable, let it be identical to the other, and add the resulting pair to the unifier; otherwise...
    2. compare their functors; if they do not match, then fail; otherwise...
    3. for each pair of respective arguments, unify the two arguments using this procedure.

Unification examples

Queen Mary
University of London

- P(x, y) unified with P(A, B) gives P(A, B) unifier { A/x, B/y }
- P(x, y) unified with Q(A, B) gives
- P(F(x)) unified with P(F(A)) gives
- P(F(x), x, u, u) unified with P(F(y), z, z, A) gives

Unification examples

- P(x, y) unified with P(A, B) gives F(A, B) unifier { A/x, B/y }
- P(x, y) unified with Q(A, B) gives no unifier (P ≠ Q)
- P(F(x)) unified with P(F(A)) gives P(F(A)) unifier { A/x }
- P(F(x), x, u, u) unified with P(F(y), z, z, A) gives

  P(F(A), A, A, A) unifier { A/x, x/y, x/z, x/u }
  - note that we don't need to write down all the different permutations
    - this is enough to say that they're all the same

Resolution example

- Some rules
  - All people who are graduating are happy. All happy people smile. Jane is graduating.
  - and a question
    - Is Jane smiling?
- First convert to predicate logic
  - Premise
    - $\forall x.(Graduating(x) \rightarrow Happy(x)) \wedge \forall x.(Happy(x) \rightarrow Smiling(x)) \wedge Graduating(Jane)$
  - Goal
    - $Smiling(Jane)$

Resolution example

- Convert to CNF
  - Premise
    - $\forall x.\text{Graduating}(x) \rightarrow \text{Happy}(x) \wedge \forall x.\text{Happy}(x) \rightarrow \text{Smiling}(x) \wedge \text{Graduating}(\text{Jane})$

1. Rewrite $\rightarrow$
    - $\forall x.(\neg\text{Graduating}(x) \vee \text{Happy}(x)) \wedge \forall x.(\neg\text{Happy}(x) \vee \text{Smiling}(x)) \wedge \text{Graduating}(\text{Jane})$
2. Reduce scope of negations: all minimal, so nothing to do
3. Rewrite double negations: no double negations so nothing to do
4. Standardise variables apart
    - $\forall x.(\neg\text{Graduating}(x) \vee \text{Happy}(x)) \wedge \forall y.(\neg\text{Happy}(y) \vee \text{Smiling}(y)) \wedge \text{Graduating}(\text{Jane})$

Resolution example

5. Skolemise existentials
    - no existentials, so nothing to do
6. Drop all universal quantifiers
    - $(\neg Graduating(x) \vee Happy(x)) \wedge (\neg Happy(y) \vee Smiling(y)) \wedge Graduating(\ Jane\ )$
7. Convert to CNF
    - already in CNF
8. Separate into disjunctive clauses
    - $\neg Graduating(x) \vee Happy(x)$
    - $\neg Happy(y) \vee Smiling(y)$
    - $Graduating(\ Jane\ )$

- Standardise variables apart between clauses
    - no change: they're already named apart

Resolution example with unification

Queen Mary University of London

- Now, the only rules we need are
  - unification
  - resolution
- Note that, in this proof, only the literals used are shown
  - but they're all there, all the time

Premises in CNF

¬Smiling(Jane)   ¬Happy(y) ∨ Smiling(y)   *Resolution*   Unifiers   { Jane/y }

Resolution   ¬Happy(Jane)   ¬Graduating(x) ∨ Happy(x)   { Jane/x }

Resolution   ¬Graduating(Jane)   Graduating(Jane)

{}

Negated goal

Example with existential quantification

- Some rules
  - All people who are graduating are happy. All happy people smile. Someone is graduating.
  - and a question
    - Is anyone smiling?
- First convert to predicate logic
  - Premise
    - $\forall x.(\text{Graduating}(x) \rightarrow \text{Happy}(x)) \wedge \forall x.(\text{Happy}(x) \rightarrow \text{Smiling}(x)) \wedge \exists x.\text{Graduating}(x)$
  - Goal
    - $\exists x.\text{Smiling}(x)$

Example with existential quantification

- Conversion to CNF
  - This time I've added the negated goal to the premises, instead of later
    - $\forall x.(Graduating(x) \rightarrow Happy(x)) \wedge \forall x.(Happy(x) \rightarrow Smiling(x)) \wedge \exists x.Graduating(x) \wedge \neg\exists x.Smiling(x)$
    - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall x.(\neg Happy(x) \vee Smiling(x)) \wedge \exists x.Graduating(x) \wedge \neg\exists x.Smiling(x)$
    - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall x.(\neg Happy(x) \vee Smiling(x)) \wedge \exists x.Graduating(x) \wedge \forall x.\neg Smiling(x)$
    - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall y.(\neg Happy(y) \vee Smiling(y)) \wedge \exists z.Graduating(z) \wedge \forall u.\neg Smiling(u)$
    - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall y.(\neg Happy(y) \vee Smiling(y)) \wedge Graduating(\ Someone\ ) \wedge \forall u.\neg Smiling(u)$
    - $(\neg Graduating(x) \vee Happy(x)) \wedge (\neg Happy(y) \vee Smiling(y)) \wedge Graduating(\ Someone\ ) \wedge \neg Smiling(u)$
    - $\{\ \neg Graduating(x) \vee Happy(x), \neg Happy(y) \vee Smiling(y), Graduating(\ Someone\ ), \neg Smiling(u)\ \}$

Example with existential quantification

- Note that the proof is exactly the same as with the constant "Jane"
  - the only difference is that the variable u gets passed around instead

$$\dfrac{\neg\text{Smiling}(u) \qquad \neg\text{Happy}(y) \lor \text{Smiling}(y)}{\neg\text{Happy}(u)}\ \textit{Resolution} \qquad \{\, u/y \,\}$$

$$\dfrac{\neg\text{Happy}(u) \qquad \neg\text{Graduating}(x) \lor \text{Happy}(x)}{\neg\text{Graduating}(u)}\ \textit{Resolution} \qquad \{\, u/x \,\}$$

$$\dfrac{\neg\text{Graduating}(u) \qquad \text{Graduating}(\text{ Someone })}{\{\}}\ \textit{Contradiction} \qquad \{\, \text{Someone}/u \,\}$$

Example with universal quantification

- Some rules
  - All people who are graduating are happy. All happy people smile. Everyone is graduating.
  - and a question
    - Is everyone smiling?
- First convert to predicate logic
  - Premise
    - $\forall x.(Graduating(x) \rightarrow Happy(x)) \wedge \forall x.(Happy(x) \rightarrow Smiling(x)) \wedge \forall x.Graduating(x)$
  - Goal
    - $\forall x.Smiling(x)$

Example with universal quantification

- Conversion to CNF
  - $\forall x.(Graduating(x) \rightarrow Happy(x)) \wedge \forall x.(Happy(x) \rightarrow Smiling(x)) \wedge \forall x.Graduating(x) \wedge \neg\forall x.Smiling(x)$
  - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall x.(\neg Happy(x) \vee Smiling(x)) \wedge \forall x.Graduating(x) \wedge \neg\forall x.Smiling(x)$
  - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall x.(\neg Happy(x) \vee Smiling(x)) \wedge \forall x.Graduating(x) \wedge \exists x.\neg Smiling(x)$
  - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall y.(\neg Happy(y) \vee Smiling(y)) \wedge \forall z.Graduating(z) \wedge \exists u.\neg Smiling(u)$
  - $\forall x.(\neg Graduating(x) \vee Happy(x)) \wedge \forall y.(\neg Happy(y) \vee Smiling(y)) \wedge \forall z. Graduating(z) \wedge \neg Smiling(Someone)$
  - $(\neg Graduating(x) \vee Happy(x)) \wedge (\neg Happy(y) \vee Smiling(y)) \wedge Graduating(z) \wedge \neg Smiling(Someone)$
  - $\{ \neg Graduating(x) \vee Happy(x), \neg Happy(y) \vee Smiling(y), Graduating(z), \neg Smiling(Someone) \}$

Example with universal quantification

Queen Mary
University of London

- Note that the proof is exactly the same as with the constant Jane
  - the difference is that the Skolem constant Someone gets passed around instead

¬Smiling(Someone)   ¬Happy(y) ∨ Smiling(y)   { Someone/y }

Resolution

———————————————————————

¬Happy(Someone)   ¬Graduating(x) ∨ Happy(x)   { Someone/x }

Resolution

———————————————————————

¬Graduating(Someone)   Graduating(z)   { Someone/z }

Contradiction

———————————————————————

{}

Alternative proof

Queen Mary
University of London

- Here, we use general resolution first – but the effect is the same

$$\dfrac{\neg\mathrm{Graduating}(x) \lor \mathrm{Happy}(x) \qquad \neg\mathrm{Happy}(y) \lor \mathrm{Smiling}(y)}{}\ \textit{Resolution} \qquad \{\ x/y\ \}$$

$$\textit{Resolution}\ \dfrac{\neg\mathrm{Graduating}(x) \lor \mathrm{Smiling}(x) \qquad \mathrm{Graduating}(\ z\ )}{} \qquad \{\ x/z\ \}$$

$$\textit{Contradiction}\ \dfrac{\mathrm{Smiling}(x) \qquad \neg\mathrm{Smiling}(\ \mathrm{Someone}\ )}{\{\}} \qquad \{\ \mathrm{Someone}/x\ \}$$

A (slightly) more realistic example

Queen Mary University of London

- In this example, not all of the quantifiers match neatly up
  - Some rules and a question
    - All people who are graduating are happy. All happy people smile. Someone is graduating.
    - Is everyone smiling?
- First convert to predicate logic
    - $\forall x.(Graduating(x) \rightarrow Happy(x)) \wedge \forall x.(Happy(x) \rightarrow Smiling(x)) \wedge \exists x.Graduating(x)$
    - $\forall x.Smiling(x)$

Skolem constants

- Resolution-ready form
    - $\neg Graduating(x) \vee Happy(x), \neg Happy(y) \vee Smiling(y) \wedge Graduating(Z) \wedge \neg Smiling(U)$

A (slightly) more realistic example

- We can't infer that everyone is smiling from the knowledge that one person is graduating

$$\dfrac{\neg Smiling(\ U\ ) \qquad \neg Happy(y) \vee Smiling(y)}{\neg Happy(\ U\ )}\ \textit{Resolution} \qquad \{\ U/y\ \}$$

$$\dfrac{\neg Happy(\ U\ ) \qquad \neg Graduating(x) \vee Happy(x)}{\neg Graduating(\ U\ )}\ \textit{Resolution} \qquad \{\ U/x\ \}$$

$$\dfrac{\neg Graduating(\ U\ ) \qquad Graduating(\ Z\ )}{\textit{No proof}} \qquad \textit{No unifier}$$

The other way round

- In this example, not all of the quantifiers match neatly up
  - Some rules and a question
    - All people who are graduating are happy. All happy people smile. Everyone is graduating.
    - Is anyone smiling?
- First convert to predicate logic
    - $\forall x.(\text{Graduating}(x) \rightarrow \text{Happy}(x)) \land \forall x.(\text{Happy}(x) \rightarrow \text{Smiling}(x)) \land \exists x.\text{Graduating}(x)$
    - $\forall x.\text{Smiling}(x)$
- Resolution-ready form
    - $\neg\text{Graduating}(x) \lor \text{Happy}(x), \neg\text{Happy}(y) \lor \text{Smiling}(y) \land \text{Graduating}(z) \land \neg\text{Smiling}(u)$

The other way round

- We can infer that one is person is smiling from the knowledge that everyone is graduating

$\neg$Smiling(u) $\qquad$ $\neg$Happy(y) $\lor$ Smiling(y) $\qquad$ { u/y }

——————————————— *Resolution*

Resolution $\quad$ $\neg$Happy(u) $\qquad$ $\neg$Graduating(x) $\lor$ Happy(x) $\qquad$ { u/x }

———————————————

Contradiction $\quad$ $\neg$Graduating(u) $\qquad$ Graduating(z) $\qquad$ { u/z }

———————————————

{}

Something to try

- Express the following sentences in FOPC
  - Every man who owns a donkey beats it.
  - John looked at Jane in the park with the telescope.
  - All lecturers except Geraint are boring.