
Artificial Intelligence

III Knowledge and Reasoning

III.5 Resolution

F. C. Langbein

School of Computer Science
and Informatics
Cardiff University





Overview

- Resolution
- Gödel's incompleteness theorem

Completeness in FOL

- Procedure \perp is *complete* iff $KB \vdash_{\perp} \alpha$ whenever $KB \models \alpha$
- Example:
 - $\forall x \text{ PhD}(x) \Rightarrow \text{HighlyQualified}(x)$
 - $\forall x \neg \text{PhD}(x) \Rightarrow \text{EarlyEarnings}(x)$
 - $\forall x \text{ HighlyQualified}(x) \Rightarrow \text{Rich}(x)$
 - $\forall x \text{ EarlyEarnings}(x) \Rightarrow \text{Rich}(x)$
 - ➔ Should be able to infer $\text{Rich}(Me)$!?
- Forward and backward chaining **not complete for general first order logic**



Gödel's Theorem

- Let F be a formal axiomatic system that is powerful enough to do (integer) arithmetic
 - In particular it has to contain the induction axiom
- Then it is possible to construct a “Gödel sentence” $G(F)$ with the following properties
 - $G(F)$ can be written as a sentence in F
 - $G(F)$ cannot be proved using the axioms in F
 - but if F is consistent, then $G(F)$ is true
- “There are true sentences that cannot be proved” (incompleteness theorem, Kurt Gödel, 1931)
 - Also see Turing and Church, 1936 (e.g. halting problem)

Sketch of Gödel's Proof

- Integer arithmetic in logic:
constant 1 and successor function $S(x)$
 - $1 = 1, 2 = S(1), 3 = S(S(1)), \dots$
 - Also add $+, \times, \exp$ and the usual FOL symbols
- Enumerate all sentences in this language:
 - Assign an alphabetical order on the symbols and sort sentences of length 1, 2, ...
 - Each sentence α gets a number $\# \alpha$ (Gödel number)
- ➔ Number theory contains a name for each of its own sentences!
- ➔ Each possible proof P has a Gödel number $G(P)$ as it is a finite sequence of sentences

Sketch of Gödel's Proof

- Let A be a set of enumerable, true sentences about natural numbers
 - A is named by a given set of integers
- Let $\alpha(k, A)$ be the sentence:
 $\forall l$ l is not the Gödel number of a proof of the sentence with Gödel number k , where the proof uses only premises in A
 - Let σ be $\alpha(\# \sigma, A)$ (σ says that σ is not provable from A)
- Assume that σ is provable from A
 - Then σ is false, so A does not consist of true sentences
 - This violates premise for A , so σ is not provable from A
- But that's what σ says! Hence, σ is true

QED.

FOL Resolution

- Entailment in first-order logic is only **semi-decidable**:
 - Can find a proof of α if $KB \models \alpha$
 - Cannot always prove that $KB \not\models \alpha$
- **Resolution** (indirect proof):
 - To prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable
 - Resolution uses $KB, \neg\alpha$ in **CNF**
 - Resolution rule combines two clauses to make new one
 - Inference continues until an **empty clause** is derived (contradiction)
- Resolution: complete in PL and refutation-complete in FOL (cannot generate all logical consequences, but can establish that a given sentence is entailed by a KB)

FOL Resolution Rule

➤ Basic propositional version

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma} \equiv \frac{\neg\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

➤ Full first-order version

$$\begin{array}{c} p_1 \vee \dots \vee p_j \vee \dots \vee p_m, \\ q_1 \vee \dots \vee q_k \vee \dots \vee q_n \end{array}$$

$$(p_1 \vee \dots \vee p_{j-1} \vee p_{j+1} \dots \vee p_m \vee q_1 \dots \vee q_{k-1} \vee q_{k+1} \dots \vee q_n) \sigma$$

where $p_j \sigma = \neg q_k \sigma$

➤ Example:

$$\frac{\neg\text{Rich}(x) \vee \text{Unhappy}(x), \text{Rich}(\text{Me})}{\text{Unhappy}(\text{Me})} \quad \text{with } \sigma = \{x/\text{Me}\}$$

Conjunctive Normal Form

- Literal = (possibly negated) atomic sentence, e.g. $\neg\text{Rich}(\text{Me})$
- Clause = disjunction of literals, e.g. $\neg\text{Rich}(\text{Me}) \vee \text{Unhappy}(\text{Me})$
- The *KB* is a conjunction of clauses
- Any FOL *KB* can be *converted to CNF* as follows:
 - Replace $P \Leftrightarrow Q$ by $P \Rightarrow Q$ and $Q \Rightarrow P$
 - Replace $P \Rightarrow Q$ by $\neg P \vee Q$
 - Move \neg inwards, e.g. $\neg\forall x P$ becomes $\exists x \neg P$
 - Standardise variables apart, e.g. $\forall x P \vee \exists x Q$ becomes $\forall x P \vee \exists y Q$
 - Move quantifiers left in order, e.g. $\forall x P \vee \exists y Q$ becomes $\forall x \exists y P \vee Q$
 - Eliminate \exists by skolemisation (next slide)
 - Drop universal quantifiers
 - Distribute \wedge over \vee , e.g. $(P \wedge Q) \vee R$ becomes $(P \vee R) \wedge (Q \vee R)$

Skolemisation

- $\exists x \text{ Rich}(x)$ becomes $\text{Rich}(G1)$ where $G1$ is a new “skolem constant”
- Tricky when \exists is inside \forall
 - Example: $\forall x \text{ Person}(x) \Rightarrow \exists y \text{ Heart}(y) \wedge \text{Has}(x, y)$
 - **Incorrect:** $\forall x \text{ Person}(x) \Rightarrow \text{Heart}(H1) \wedge \text{Has}(x, H1)$

Skolemisation

- $\exists x \text{ Rich}(x)$ becomes $\text{Rich}(G1)$ where $G1$ is a new “**skolem constant**”
- Tricky when \exists is inside \forall
 - Example: $\forall x \text{ Person}(x) \Rightarrow \exists y \text{ Heart}(y) \wedge \text{Has}(x, y)$
 - **Incorrect**: $\forall x \text{ Person}(x) \Rightarrow \text{Heart}(H1) \wedge \text{Has}(x, H1)$
 - **Correct**: $\forall x \text{ Person}(x) \Rightarrow \text{Heart}(H(x)) \wedge \text{Has}(x, H(x))$
where H is a new symbol: “**Skolem function**”
- Skolem function arguments:
all *enclosing* universally quantified variables

CNF Conversion Example

➤ $\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{ Loves}(y, x)]$
 (“Everyone who loves all animals is loved by someone”)

⋮
⋮
⋮
⋮
⋮

➤ Gives:

- $\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)$
- $\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)$
 - with skolem functions F, G

Resolution Proof

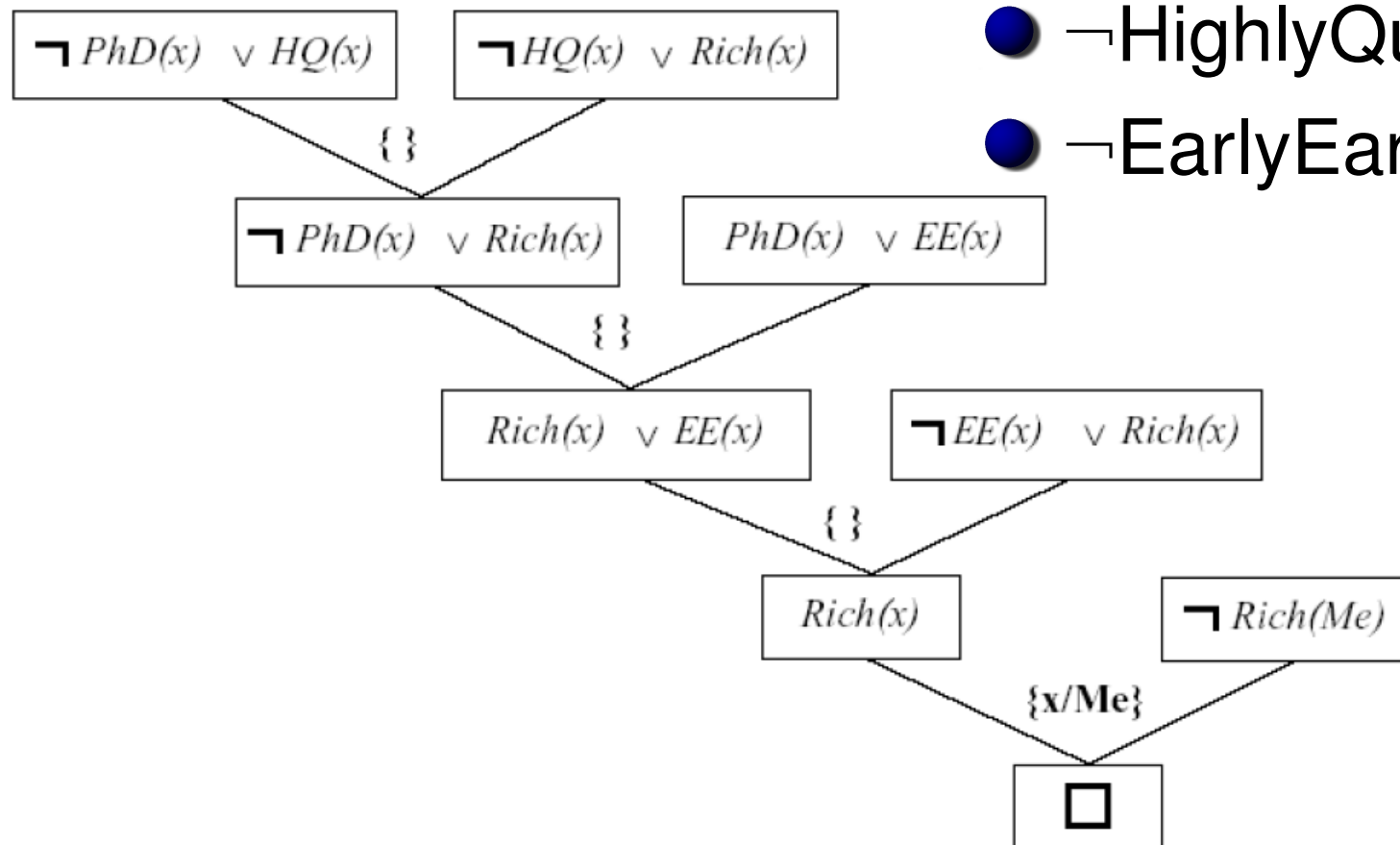
- To proof α given knowledge base KB :
 - Negate α : $\neg\alpha$
 - Add $\neg\alpha$ to KB
 - Convert KB into CNF
 - Infer contradiction (empty clause) using resolution rule

```
function result  $\leftarrow$  RESOLUTION ( $KB, \alpha$ )  
clauses  $\leftarrow$  CNF ( $KB \wedge \neg\alpha$ )  
while false  $\notin$  clauses  
    ( $S_1, S_2$ )  $\leftarrow$  SELECTUNIFIABLE (clauses)  
    if ( $S_1, S_2$ ) = empty, return false  
    resolvents  $\leftarrow$  RESOLVE ( $S_1, S_2$ )  
    clauses  $\leftarrow$  clauses  $\cup$  resolvents
```

Rich(Me) Resolution Example

➤ Example: to prove Rich(Me),
add \neg Rich(Me) to the CNF *KB*:

- \neg PhD(x_1) \vee HighlyQualified(x_1)
- PhD(x_2) \vee EarlyEarnings(x_2)
- \neg HighlyQualified(x_3) \vee Rich(x_3)
- \neg EarlyEarnings(x_4) \vee Rich(x_4)



Criminal(West) Resolution Example

