Foundations of Artificial Intelligence

CS472/3 — Fall 1999
Lecture #14 & 15

Bart Selman

Today's Lecture

Knowledge Representation & Reasoning
Chapter 6 & 7, R&N.

Logical Representation

Three components:

syntax
semantics (link to the world)
proof theory ("pushing symbols")

To make it work: **soundness** and **completeness**.

Soundness and Completeness

Allows computer to ignore semantics and "just push symbols"!
In propositional logic, truth tables cumbersome (at least).
In first-order, models can be infinite!

Proof theory: One or more **inference rules** with
zero or more **axioms** (tautologies / to get things "going")

Example Proof Theory

One rule of inference: **Modens Ponens**

From α and $\alpha \Rightarrow \beta$ it follows that β.

Semantic soundness easily verified. (truth table)

Axiom schemas:

(Ax. I) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

(Ax. II) $((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)))$.

(Ax. III) $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\alpha \Rightarrow \neg\beta) \Rightarrow \alpha$.

Note: α, β, γ stand for arbitrary sentences. So, infinite collection of axioms.

Slide CS472–5

Example Proof

Lemma. 1) For any α, we have $\vdash (\alpha \Rightarrow \alpha)$.

Proof.

$(\alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha \Rightarrow \alpha) \Rightarrow \alpha \Rightarrow \alpha$, (Ax. II)

$\alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha$, (Ax. I)

$(\alpha \Rightarrow \alpha \Rightarrow \alpha) \Rightarrow \alpha \Rightarrow \alpha$, (M. P.)

$\alpha \Rightarrow \alpha \Rightarrow \alpha)$ (Ax. I)

$\alpha \Rightarrow \alpha$ (M.P.)

Slide CS472–6

Another Example Proof

Lemma. 2) For any α and β, we have $\beta, \neg\beta \vdash \alpha$.

Proof.

$(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\alpha \Rightarrow \neg\beta) \quad \vdash \quad \alpha$, (Ax. III)

β, (hyp.)

$\beta \Rightarrow \neg\alpha \Rightarrow \beta$, (Ax. I)

$\neg\alpha \Rightarrow \beta$, (M.P.)

$(\neg\alpha \Rightarrow \neg\beta) \Rightarrow \alpha$, (M.P.)

$\neg\beta$, (hyp.)

$\neg\beta \Rightarrow \neg\alpha \Rightarrow \neg\beta$, (Ax. I)

$\neg\alpha \Rightarrow \neg\beta$, (M.P.)

α (M.P.)

Slide CS472–7

Q. Why are lemma 1 and 2 true semantically?

I.e., $\models \alpha \Rightarrow \alpha$ and $\beta, \neg\beta \models \alpha$.

Note: **proofs** purely **syntactic** — machine does not need to know anything about the meaning of the sentences!

Whatever is syntactically derived will be semantically true, and we can derive everything syntatically that is semantically true.

How hard is it to find proofs?

Slide CS472–8

Key Properties

We have the following properties (also for first-order logic):

the following three conditions are equivalent:

(I) $\Phi \models \alpha$

(II) $\Phi \vdash \alpha$

(III) $\Phi, \neg\alpha$ is inconsistent (can be refuted).

(I) is semantic; (II) syntactic, and (III) at high-level semantic

but we have a nice syntactic automatic procedure procedure: **resolution**.

What common proof techinque does III represent?

Slide CS472–9

Resolution

First need canonical form: "clausal".

Conjunction of disjunctions (clauses) /CNF (conj. norm. form)

Ex.: $\neg(P \Rightarrow Q) \vee (R \Rightarrow P)$.

$\neg(\neg P \vee Q) \vee (\neg R \vee P)$

$(P \wedge \neg Q) \vee (\neg R \vee P)$ (Morgan's law)

$(P \vee \neg R \vee P) \wedge (\neg Q \vee \neg R \vee P)$ (assoc. and distr. laws)

$(P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$

$\{(P \vee \neg R), (\neg Q \vee \neg R \vee P)\}$,

$\{\{P, \neg R\}, \{\neg Q, \neg R, P\}\}$ (just notation)

Slide CS472–10

What can you say about the length of the CNF?

Given a CNF, a **single** inference rule (and no axioms) will allow us to determine inconsistency.

So, using property III (above) and resolution, we have a sound and complete proof procedure for propositional logic (can be extended to first-order).

The Resolution Rule (clausal form)

From $\alpha \vee p$ and $\neg p \vee \beta$, we can derive:
$\alpha \vee \beta$ (α and β are disjunctions of literals (literal = prop. vars or its negation).

Note: $\neg\alpha \Rightarrow p$ and $p \Rightarrow \beta$ gives $\neg\alpha \Rightarrow \beta$.

It's a **"chaining rule."**

We can derive the empty clause via resolution iff
the set of clauses is inconsistent.

Method relies on property III. It's **refutation complete.**
Note that method does not generate theorems from scratch.
E.g. we have $P \wedge R \models (P \vee R)$, but we can't get
$(P \vee R)$ from $\{\{P\}, \{R\}\}$. (Why not??)

But, given $\{\{P\}, \{R\}\}$ and the negation of $P \vee R$, we
get the set $\{\{P\}, \{R\}, \{\neg P\}, \{\neg R\}\}$. Resolving
on this set gives empty clause. Thus contradiction.
Thus proof.

Slide CS472–13

Example.
1) $ARM-OK$
2) $\neg MOVES$
3) $ARM-OK \wedge LIFTABLE \Rightarrow MOVES$ (i.e.)
4) $\neg ARM-OK \vee \neg LIFTABLE \vee MOVES$)
Prove: $\neg LIFTABLE$.

5) $LIFTABLE$ (assert)
6) $\neg ARM-OK \vee MOVES$ (resolving 5 and 4)
7) $\neg ARM-OK$ (from 6 and 2)
8) Nil (empty clause / contradiction, from 7 and 1).

Slide CS472–14

May seem cumbersome but note that can be easily automated. Just “smash” clauses till empty clause or no more new clauses.

Guaranteed sound and (refutation) complete.

Q. Why is method with axioms more difficult to implement?

Slide CS472–15

What about length of resolution proof?

Consider Pigeon-Hole (PH) problem: Formula encodes that you cannot place $n+1$ pigeons in n holes (one per hole).

Cook / Karp around 1971/72. “Resolved” by Armin Haken 1985.
Related to NP vs. $co-NP$ questions.
PH takes **exponentially** many steps! (no matter in what order.)

PH hidden in many practical problems. Makes thm. proving expensive. Partly, led to recent move to model-based methods (NP-complete).

Slide CS472–16

Pigeon-Hole Principle

$P_{i,j}$ for Pigeon i in hole j.

$P_{1,1} \lor P_{1,2} \lor P_{1,3} \ldots P_{1,n}$

$P_{2,1} \lor P_{2,2} \lor P_{2,3} \ldots P_{2,n}$

$\ldots$

$P_{(n+1),1} \lor P_{(n+2),2} \lor P_{(n+3),3} \ldots P_{(n+1),n}$

and ??

Slide CS472–17

$(\neg P_{1,1} \lor \neg P_{1,2})$, $(\neg P_{1,1} \lor \neg P_{1,3})$, $(\neg P_{1,1} \lor \neg P_{1,4})$

$\ldots$

$(\neg P_{1,(n-1)} \lor \neg P_{1,n})$,

$(\neg P_{2,1} \lor \neg P_{2,2}) \ldots (\neg P_{2,(n-1)} \lor \neg P_{2,n})$

etc.

$(\neg P_{1,1} \lor \neg P_{2,1})$, $(\neg P_{1,1} \lor \neg P_{3,1})$, $\ldots$

$(\neg P_{1,2} \lor \neg P_{2,2})$, $(\neg P_{1,2} \lor \neg P_{3,2})$, etc.

Slide CS472–18

Resolution proof of inconsistency requires at least an exponential number of clauses, no matter in what order how you resolve things!

"Method can't count."

Slide CS472–19

A More Concise Formulation

$\forall x \exists y (x \in Pigeons)(y \in Holes) IN(x, y)$

$\forall x \forall x' \forall y (IN(x, y) \wedge IN(x', y) \ldots ??$

$\forall x \forall y \forall y' (IN(x, y) \wedge IN(x, y') \ldots ??$

$Pigeons = \{p_1, p_2, ...p_{n+1}\}$,

$Holes = \{p_1, p_2, ...p_n\}$.

We have **first-order logic** with some set-theory notation.

Notation only.

Alternatively, we can state for $x \in Pigeons$ as ??

Q. Any easier to determine inconsistency?

Slide CS472–20

First-order logic

Will give us a more concise formulation.

For finite domains, essentially equiv. to propositional logic. Often pays off to expand!

First-order: extends propositional with variables, predicate symbols, functions, and quantifiers ($\exists, \forall$).

We'll see that key properties from propositional case carry over:

Model theoretic semantics; sound and complete proof theory; sound and refutation complete resolution.