Logic: Resolution Proofs; Objects and Relations

CPSC 322 Lecture 21

March 7, 2007
Textbook §4.2

Lecture Overview

Proofs

- A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.
- Given a proof procedure, $KB \vdash g$ means g can be derived from knowledge base KB.
- Recall $KB \models g$ means g is true in all models of KB.

Definition (soundness)

A proof procedure is sound if $KB \vdash g$ implies $KB \models g$.

Definition (completeness)

A proof procedure is complete if $KB \models g$ implies $KB \vdash g$.

Bottom-up Ground Proof Procedure

One rule of derivation, a generalized form of *modus ponens*:

> *If "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" is a clause in the knowledge base, and each b_i has been derived, then h can be derived.*

You are forward chaining on this clause.
(This rule also covers the case when $m = 0$.)

Soundness of bottom-up proof procedure

If $KB \vdash g$ then $KB \models g$.

- Suppose there is a g such that $KB \vdash g$ and $KB \not\models g$.
- Let h be the first atom added to C that's not true in every model of KB.
- Suppose h isn't *true* in model I of KB.
- There must be a clause in KB of form

  $$h \leftarrow b_1 \wedge \ldots \wedge b_m$$

  Each b_i is true in I. h is false in I. So this clause is false in I.
- Therefore I isn't a model of KB. Contradiction: thus no such g exists.

Minimal Model

We can use proof procedure to find a model of KB.

- First, observe that the C generated at the end of the bottom-up algorithm is a fixed point.
  - further applications of our rule of derivation will not change C.

Let I be the interpretation in which every element of the fixed point C is true and every other atom is false.

- we'll call I a minimal model.

Claim: I is a model of KB. Proof:

- Assume that I is not a model of KB. Then there must exist some clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ in KB (having zero or more b_i's) which is false in I.
- This can only occur when h is false and each b_i is true in I.
- If each b_i belonged to C, we would have added h to C as well.
- Since C is a fixed point, no such I can exist.

Completeness

If $KB \models g$ then $KB \vdash g$.

- Suppose $KB \models g$. Then g is true in all models of KB.
- Thus g is true in the minimal model.
- Thus g is generated by the bottom up algorithm.
- Thus $KB \vdash g$.

Lecture Overview

Top-down Ground Proof Procedure

Idea: search backward from a query to determine if it is a logical consequence of KB.

An answer clause is of the form:

$$yes \leftarrow a_1 \land a_2 \land \ldots \land a_m$$

The SLD Resolution of this answer clause on atom a_i with the clause:

$$a_i \leftarrow b_1 \land \ldots \land b_p$$

is the answer clause

$$yes \leftarrow a_1 \land \cdots \land a_{i-1} \land b_1 \land \cdots \land b_p \land a_{i+1} \land \cdots \land a_m.$$

Derivations

- An answer is an answer clause with $m = 0$. That is, it is the answer clause $yes \leftarrow$.
- A derivation of query "$?q_1 \wedge \ldots \wedge q_k$" from KB is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  - γ_0 is the answer clause $yes \leftarrow q_1 \wedge \ldots \wedge q_k$,
  - γ_i is obtained by resolving γ_{i-1} with a clause in KB, and
  - γ_n is an answer.

Top-down definite clause interpreter

To solve the query $?q_1 \wedge \ldots \wedge q_k$:

> $ac :=$ "$yes \leftarrow q_1 \wedge \ldots \wedge q_k$"
> **repeat**
> > **select** atom a_i from the body of ac;
> > **choose** clause C from KB with a_i as head;
> > replace a_i in the body of ac by the body of C
>
> **until** ac is an answer.

- Don't-care nondeterminism If one selection doesn't lead to a solution, there is no point trying other alternatives. select
- Don't-know nondeterminism If one choice doesn't lead to a solution, other choices may. choose

Example: successful derivation

$$
\begin{array}{lll}
a \leftarrow b \wedge c. & a \leftarrow e \wedge f. & b \leftarrow f \wedge k. \\
c \leftarrow e. & d \leftarrow k. & e. \\
f \leftarrow j \wedge e. & f \leftarrow c. & j \leftarrow c.
\end{array}
$$

Query: $?a$

$$
\begin{array}{llll}
\gamma_0: & yes \leftarrow a & \gamma_4: & yes \leftarrow e \\
\gamma_1: & yes \leftarrow e \wedge f & \gamma_5: & yes \leftarrow \\
\gamma_2: & yes \leftarrow f & & \\
\gamma_3: & yes \leftarrow c & &
\end{array}
$$

Example: failing derivation

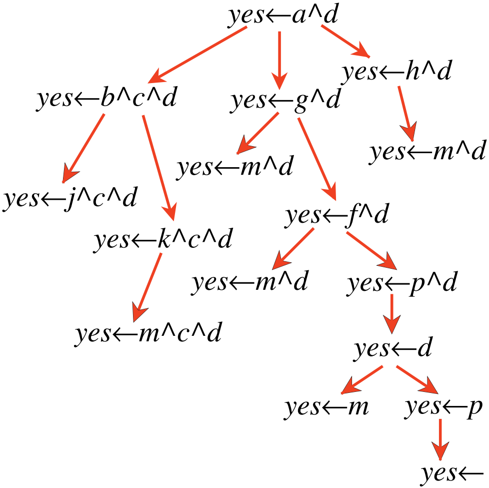
$$
\begin{array}{lll}
a \leftarrow b \wedge c. & a \leftarrow e \wedge f. & b \leftarrow f \wedge k. \\
c \leftarrow e. & d \leftarrow k. & e. \\
f \leftarrow j \wedge e. & f \leftarrow c. & j \leftarrow c.
\end{array}
$$

Query: $?a$

$$
\begin{array}{ll}
\gamma_0 : & yes \leftarrow a \\
\gamma_1 : & yes \leftarrow b \wedge c \\
\gamma_2 : & yes \leftarrow f \wedge k \wedge c \\
\gamma_3 : & yes \leftarrow c \wedge k \wedge c \\
\gamma_4 : & yes \leftarrow e \wedge k \wedge c \\
\gamma_5 : & yes \leftarrow k \wedge c
\end{array}
$$

Search Graph for SLD Resolution

$$
\begin{array}{ll}
a \leftarrow b \wedge c. & a \leftarrow g. \\
a \leftarrow h. & b \leftarrow j. \\
b \leftarrow k. & d \leftarrow m. \\
d \leftarrow p. & f \leftarrow m. \\
f \leftarrow p. & g \leftarrow m. \\
g \leftarrow f. & k \leftarrow m. \\
h \leftarrow m. & p. \\
?a \wedge d &
\end{array}
$$



Lecture Overview

Objects and Relations

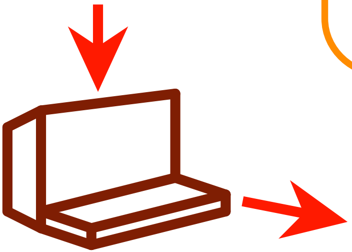
- It is useful to view the world as consisting of objects and relationships between these objects.
- Often the propositions we spoke about before can be condensed into a much smaller number of propositions if they are allowed to express relationships between objects and/or functions of objects.
- Thus, reasoning in terms of objects and relationships can be simpler than reasoning in terms of features, as you can express more general knowledge using logical variables.

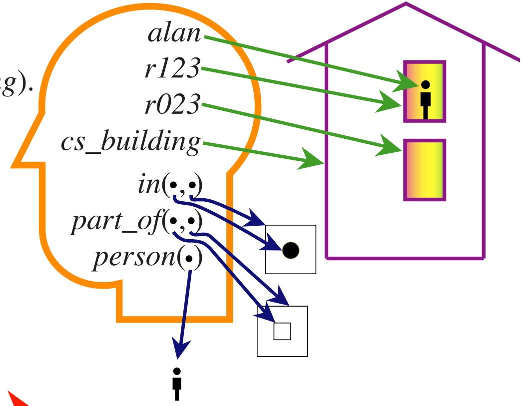
Using an RRS

1. Begin with a task domain.
2. Distinguish those objects you want to talk about.
3. Determine what relationships you want to represent.
4. Choose symbols in the computer to denote objects and relations.
5. Tell the system knowledge about the domain.
6. Ask the system questions.

Example Domain for an RRS

in(alan,r123).
part_of(r123,cs_building).
in(X,Y) ← part_of(Z,Y) ∧ in(X,Z).





in(alan,cs_building)

Representational Assumptions of Datalog

- An agent's knowledge can be usefully described in terms of *individuals* and *relations* among individuals.
- An agent's knowledge base consists of *definite* and *positive* statements.
- The environment is *static*.
- There are only a finite number of individuals of interest in the domain. Each individual can be given a unique name.

$\Longrightarrow$ Datalog