Logic: Objects and Relations

CPSC 322 Lecture 22

March 6, 2006
Textbook Chapter 5

Lecture Overview

Recap

Datalog

Datalog Syntax

Semantics of Datalog

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:

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

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

$$a_i \leftarrow b_1 \wedge \ldots \wedge b_p$$

is the answer clause

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

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 $?q_1 \land \ldots \land q_k$:

> $ac :=$ "$yes \leftarrow q_1 \land \ldots \land q_k$"
> **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.

Lecture Overview

Recap

Datalog

Datalog Syntax

Semantics of Datalog

Objects and Relations

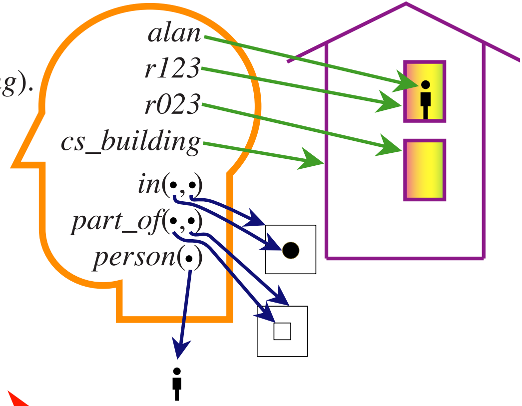
- It is useful to view the world as consisting of objects and relationships between these objects.
- Often the propositions we spoke about before can be condensed into a much smaller number of propositions if they are allowed to express relationships between objects and/or functions of objects.
- Thus, 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.

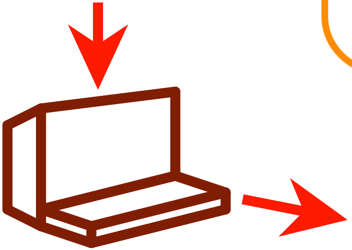
Using an RRS

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

Lecture Overview

Syntax of Datalog

- variable starts with upper-case letter.
- constant starts with lower-case letter or is a sequence of digits (numeral).
- predicate symbol starts with lower-case letter.
- term is either a variable or a constant.
- atomic symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.

Syntax of Datalog (cont)

- definite clause is either an atomic symbol (a fact) or of the form:

$$\underbrace{a}_{\text{head}} \quad \leftarrow \quad \underbrace{b_1 \wedge \cdots \wedge b_m}_{\text{body}}$$

  where a and b_i are atomic symbols.
- query is of the form $?b_1 \wedge \cdots \wedge b_m$.
- knowledge base is a set of definite clauses.

Example Knowledge Base

$$
\begin{aligned}
& in(alan, R) \leftarrow \\
& \qquad teaches(alan, cs322) \wedge \\
& \qquad in(cs322, R). \\
& grandfather(william, X) \leftarrow \\
& \qquad father(william, Y) \wedge \\
& \qquad parent(Y, X). \\
& slithy(toves) \leftarrow \\
& \qquad mimsy \wedge borogroves \wedge \\
& \qquad outgrabe(mome, Raths).
\end{aligned}
$$

Lecture Overview

Semantics: General Idea

- Recall: a semantics specifies the meaning of sentences in the language.
  - ultimately, we want to be able to talk about which sentences are true and which are false
- In propositional logic, all we needed to do in order to come up with an interpretation was to assign truth values to atoms
- For Datalog, an interpretation specifies:
  - what objects (individuals) are in the world
  - the correspondence between symbols in the computer and objects & relations in world
    - which constants denote which individuals
    - which predicate symbols denote which relations (and thus, along with the above, which sentences will be true and which will be false)

Formal Semantics

An interpretation is a triple $I = \langle D, \phi, \pi \rangle$, where

- D, the domain, is a nonempty set. Elements of D are individuals.
- ϕ is a mapping that assigns to each constant an element of D. Constant c denotes individual $\phi(c)$.
- π is a mapping that assigns to each n-ary predicate symbol a relation: a function from D^n into $\{$*TRUE*, *FALSE*$\}$.

Example Interpretation

Constants: $phone$, $pencil$, $telephone$.
Predicate Symbol: $noisy$ (unary), $left\_of$ (binary).

- $D = \{$✂, ☎, ✎$\}$.
- $\phi(phone) =$ ☎, $\phi(pencil) =$ ✎, $\phi(telephone) =$ ☎.
- $\pi(noisy)$:

⟨✂⟩	*FALSE*	⟨☎⟩	*TRUE*	⟨✎⟩	*FALSE*

  $\pi(left\_of)$:

⟨✂, ✂⟩	*FALSE*	⟨✂, ☎⟩	*TRUE*	⟨✂, ✎⟩	*TRUE*
⟨☎, ✂⟩	*FALSE*	⟨☎, ☎⟩	*FALSE*	⟨☎, ✎⟩	*TRUE*
⟨✎, ✂⟩	*FALSE*	⟨✎, ☎⟩	*FALSE*	⟨✎, ✎⟩	*FALSE*

Important points to note

- The domain D can contain real objects. (e.g., a person, a room, a course). D can't necessarily be stored in a computer.
- The constants do not have to match up one-to-one with members of the domain. Multiple constants can refer to the same object, and some objects can have no constants that refer to them.
- $\pi(p)$ specifies whether the relation denoted by the n-ary predicate symbol p is true or false for each n-tuple of individuals.
- If predicate symbol p has no arguments, then $\pi(p)$ is either *TRUE* or *FALSE*.

Truth in an interpretation

A constant c denotes in I the individual $\phi(c)$.
Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

- true in interpretation I if $\pi(p)(t'_1, \ldots, t'_n) =$ *TRUE*, where t_i denotes t'_i in interpretation I and
- false in interpretation I if $\pi(p)(t'_1, \ldots, t'_n) =$ *FALSE*.

Ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ is false in interpretation I if h is false in I and each b_i is true in I, and is true in interpretation I otherwise.

Example Truths

In the interpretation given before:

$noisy(phone)$	true
$noisy(telephone)$	true
$noisy(pencil)$	false
$left\_of(phone, pencil)$	true
$left\_of(phone, telephone)$	false
$noisy(pencil) \leftarrow left\_of(phone, telephone)$	true
$noisy(pencil) \leftarrow left\_of(phone, pencil)$	false
$noisy(phone) \leftarrow noisy(telephone) \wedge noisy(pencil)$	true

Models and logical consequences

- A knowledge base, KB, is true in interpretation I if and only if every clause in KB is true in I.
- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written $KB \models g$, if g is true in every model of KB.
- That is, $KB \models g$ if there is no interpretation in which KB is true and g is false.

Variables

- Variables are universally quantified in the scope of a clause.
- A variable assignment is a function from variables into the domain.
- Given an interpretation and a variable assignment, each term denotes an individual and each clause is either true or false.
- A clause containing variables is true in an interpretation if it is true for all variable assignments.