Lecture 2

CS 475

January 31, 2003

1 Introduction

1. Consider the task of writing a program for a robot that moves around the cs-building and accepts a request to go from one office to another office.

Necessary inputs:

- topological map of the cs-building (offices, topological map) where each office is associated with a name (occupant) and a number.
- a procedure *go(X,X+1)*.
- a procedure that *uses* $go(X, X+1)$ to navigate from X to Y.

3. How are we doing it?

- Encoding the map in a from of connected graph or something similar to it;

  Note: We cannot speak with the robot and make it understand using conventional communication (e.g., this is the cs-building, it has 50 offices, some of them are labs, some are professors's office, etc.) The reason: robot cannot understand!
- write a program that *corresponds* to $go(X, X+1)$;
- provide a way for users to enter two numbers

4. Refine:

- add one more procedure for checking for valid office numbers
- add a procedure that allows users to enter name, check for validity of names;

Summarize:

- We use numbers/string/graph to tell the robot to go from one place to another place;
- The world of the robot and our world is very different; yet, the robot does exactly what we ask for as though it 'understands' our request.

5. We have to communicate with the robot through a *formal language*. We *teach* it how to go from one room to another room. We also reinterpret its outputs.

6. A language has its syntax and semantics. It also has rules that provide a way for deriving new conclusions from old one.

7. A program needs to understand the syntax (parser) and often consists of procedures that reason from the input(s) to get to the output.

2 Representation and Reasoning System (RRS)

2.1 Features of a RRS

1. A domain of interest
2. Objects of interest
3. Representation of the domain (associated objects with symbols), and of knowledge about the domain
4. Tell the robot about the domain (the representation/knowledge)
5. Query the RRS about the domain

2.2 Components of a RRS

1. A formal language (legal sentences that can be used to express knowledge about a domain): grammar; KB (Knowledge Base) is a set of sentences in the language.
2. Semantics: specifies the meaning of sentences in the language; (how to understand the object...)
3. A reasoning theory (proof procedure): specification of how an answer can be derived from a knowledge base. Usually a set of inference rules;

2.3 Components of an implementation of a RRS

1. A language parser (what is legal/not?)
2. A reasoning procedure (implementation of the reasoning theory
3. Where is the semantics??? It is not implemented; it provides the meaning to the symbols; specifies what is correct; the meaning associated to symbols is often provided by the 'modeler'; different users provide different meaning;

Figure 2.

Note 1: A RRS is associated with a *representational methodology* (a user manual telling users how to use it: How to represent a domain in its language? What can be asked?)

Note 2: The implementation uses computer symbol; The domain is the 'real-world' with objects and their relations; The domain specifier uses **logic** to associate meaning to computer symbol (e.g., 1 'means' alan; 2 'means' ...)

Use of **logic** for the specification of meaning in terms of the relationship between the symbols in a computer and the task domain.

2.4 Simplifying Assumptions of the initial RRS

Simpification allows us to build simple RRS; Slowly relaxing the assumptions to allow more general cases.

Assumptions of type A, E, and AE (agent, environment, and relationship between agent and its environment, respectively).

1. **IR** Individuals and Relations: An agent's knowledge can be usefuly described in terms of *individuals* and *relations* (Type AE).
2. **DK** Definite Knowledge: An agent's KB consists of *definite* and *positive* statements (Type A).
3. **SE** Static environment: the environment is static (Type E).
4. **FD** Finite domain: There are only a finite number of individuals of interests in the domain. Each individual can be given a unique name. (Type E).

3 Datalog

3.1 Syntax

1. Variables: words starting with a upper case letter or the underscore _;
2. Constants: words starting with a lower case letter or a digit; numerals: constants with only digits;
3. Predicate symbols: words starting with a lower case letter;
4. Terms: variables or constants;
5. Atomic symbol: (atom) either p or $p(t_1, \ldots, t_i)$ where t_i are terms and p is a predicate symbol;
6. Body: a or $a_1 \wedge a_2 \ldots \wedge a_n$ where each a_i is an atom and $n > 1$; (*conjunction*, $\wedge$ means *and*);
7. Definite clause: $a \leftarrow b$ (read as "a if b) where a (caled *head*) is an atom and b is a body;
8. Query: q? (read as: prove q) where q is a body;
9. Expression: term, atom, definite clause, or query.
10. KB: set of definite clauses.

An expression is *ground* if it does not contain a variable.

Example:

$$in(alan, R) \leftarrow teaches(alan, cs322) \wedge in(cs322, R).$$
$$grandfather(william, X) \leftarrow father(william, Y) \wedge parent(Y, X).$$
$$slithy(toves) \leftarrow mimsy \wedge borogroves \wedge outgrabe(mome, Raths).$$

4 Semantics

Specifies the meaning of the symbol (a correspondence between the symbols and the objects of the domain). Needs to talk about two things:

1. the domain (objects, relations)
2. the language (symbols)

Interpretation: The correspondence between symbols of the language and objects.

Figure 2.1

Informal meanning

4.1 Formal Semantics

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

1. D is a nonempty set *domain of individuals.*
2. ϕ a mapping that assigns to each constant an element of D.
3. π a mapping that assigns to each n-ary predicate symbol a function from D^n into [TRUE,FALSE].

ϕ: function from names into individuals; c is a constant but $\phi(c)$ is a real object in the domain;

$\pi(p)$ specifies whether the relation denotes by the n-ary predicate symbol is true or false for each tuple of individuals;

Example 2.3

Each ground term denotes an individual in an interpretation; A constant c denotes in I the individual $\phi(c)$.

Each ground atom is either true or false in an interpretation. Atom $p(t_1, \ldots, p_n)$ is true in I if $\pi(p)(t'_1, \ldots, t'_n) = TRUE$ where t'_i is the individual denoted by the term t_i, and is false in I otherwise.

Truth values of a ground clause? Table $\wedge$ and $\leftarrow$

p	q	$p \wedge q$	$p \leftarrow q$
true	true	true	true
true	false	false	true
false	true	false	false
false	false	false	true

We forgot variables?

A *variable assignment*, ρ, is a function from the set of variables into the domain (D). So, each variable denotes an individual.

Given ϕ, ρ: each term (grounded or not) denotes an individual of the domain $\Rightarrow$ we can define what is the truth value of an arbitrary atom given ϕ and ρ.

A clause is true in an interpretation if it is true for all variable assignments. (**Universal quantification**) It is false in I if there exists a variable assignment under which the clause is false.

Note: Variables are quantified over the scope of the clause. So, one variable assignment for the head and the body.

A set of clauses S is true in an interpretation I if every clause c in S is true in I.

A *model* of a set of clauses S is an interpretation M such that S is true in M.

For a knowledge base (set of clauses) KB and an atom q, q is logically followed from KB, (or a consequence of KB), (or KB entails q), denoted by $KB \models q$, if q is true in every model of KB.

4.2 User's View of Semantics

Representational methodology:

1. Choose a task domain: intended interpretation.
2. Associate constants with individuals you want to name.
3. For each relation you want to represent, associate a predicate symbol in the language.
4. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
5. Ask questions about the intended interpretation.
6. If $KB \models g$, then g must be true in the intended interpretation.

4.3 Computer's View of Semantics

1. The computer doesn't have access to the intended interpretation.
2. All it knows is the knowledge base.
3. The computer can determine if a formula is a logical consequence of KB.
4. If $KB \models g$ then g must be true in the intended interpretation.
5. If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.