Contents

Representation and Reasoning System **1**

Implementation 2

Using 2

Simplifying Assumptions of Inital RRS 2

Syntax of Datalog 2

Semantics: General Idea **3**

Formal Semantics 3

Important Things to Note 3

Truth is an Interpretation 3

Models and Logical Consequence 4

User's View of Semantics 4

Computer's View of Semantics 4

Proofs **4**

Bottom-Up Ground Proof Procedure 5

Soundness 5

Fixed Point 5

Completeness 5

Representation and Reasoning System

A Representation and Reasoning System (RRS) is made up of

- *Formal language:* specifies the legal sentences
- *Semantics*: specifies the meaning of the symbols
- *Reasoning theory or proof procedure:* nondeterministic specification of how an answer can be produced

Implementation

An implementation of an RRS consists of

- *Language parser:* maps sentences of the language into data structures
- *Reasoning procedure:* implementation of reasoning theory and search strategy

  The semantics aren't reflected in the implementation!

Using

1. Begin with a task domain
2. Distinguish those things you want to talk about (the ontology)
3. Choose symbols in the computer to denote objects and relations
4. Tell the system knowledge about the domain
5. Ask the system questions

Simplifying Assumptions of Inital RRS

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.

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
- *Definite clause* is either an atomic symbol (a fact) or of the form $a \leftarrow b_1 \vee \cdots \vee b_m$
- *Query* is of the form $?b_1 \vee \cdots \vee b_m$
- *Knowledge base* is a set of definite clauses

Semantics: General Idea

A *semantics* specifies the meaning of sentences in the language. An *interpretation* specifies

- What objects (individuals) are in the world
- The correspondence between symbols in the computer and objects and relations in the world
  - Constants denote individuals
  - Predicate symbols denote relations

Formal Semantics

- An *interpretation* is a tiple $I = \{D, \phi, \pi\}$ 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 $\{true, false\}$

Important Things to Note

- The domain D can contain real objects (e.g. a person, a room) and can't necessarily be stored in a computer
- $\pi(p)$ 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 each `true` or `false`

Truth is 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
- *False in interpretation* I if $\pi(p)(t'_1, \ldots, t'_n) = false$

Ground clase $h \leftarrow b_1 \vee \cdots \vee b_m$ is *false in interpretation* I if h is false in I and each b_i is true in I, and if *true in interpretation* I otherwise.

Models and Logical Consequence

- A knowledge base KB, is true in interpration I if and only if every clause in KB is true in I
- A *model* of a set of clauses is an interpreation 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 $KB \vDash g$ if g is true in every model of KB
- That is, $KB \vDash g$ if there is no interpretation in which KB is true and g is false

User's View of Semantics

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 classes that are true in the intended interpretation: *axiomatising the domain*
5. Ask questions about the intended interpretation
6. If $KB \vDash g$ then g must be true in the intended interpretation

Computer's View of Semantics

- The computer doesn't have access to the intended interpretation
- All it knows is the knowledge base
- The computer can determine if a formula is a logic consequence of KB
- If $KB \vDash g$ then £g£ must be true in the intended interpretation
- if $KB \nvDash g$ then there is a model of KB in which g is false
    - This could be the intended interpretation

Proofs

- A *proof* is a mechanically derivable demonstration that a formula logically follows from a knowledge base
- Given a proof proceduce $KB \vdash g$ means g can be derived from knowledge base KB
- Recall $KG \vDash g$ means g is true in all models of KB
- A proof procedure is *sound* if $KB \vdash g$ implies $KB \vDash g$
- A proof procedure is *complete* is $KB \vDash g$ implies $KB \vdash g$

Bottom-Up Ground Proof Procedure

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

> If "$h \leftarrow b_1 \vee \cdots \vee b_m$" is a caluse in the knowledge base, and each b_i has been derived, than h can be derived.

You are *forward chaining* on this clause

Soundness

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

Suppose there is a g such that $KB \vdash g$ and $KB \nvDash g$.

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 in the form

$$h \leftarrow b_1 \vee \cdots \vee b_m$$

Each b_i is true in I. h is false in I. So this clause is false in I. Therefore I isn't a odel of KB.

Contradiction: thus no such g exists.

Fixed Point

The C generated atr the end of the bottom-up algorithm is called a *fixed point*.

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

I is a model of KB.

Proof: suppose $h \rightarrow b_1 \vee \cdots \vee b_m$ in KB is false in I. Then h is false and each b_i is true in I. Thus h can be added to C. Contradiction to C being the fixed point.

I is called a *Minimal Model*.

Completeness

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

Suppose $KB \vDash g$. 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 $KB \vdash g$.