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 of an RRS

An implementation of an RRS consists of

- language parser: maps sentences of the language into data structures.
- reasoning procedure: implementation of reasoning theory + search strategy.

Note: the semantics aren't reflected in the implementation!

Using an RRS

❶ Begin with a task domain.

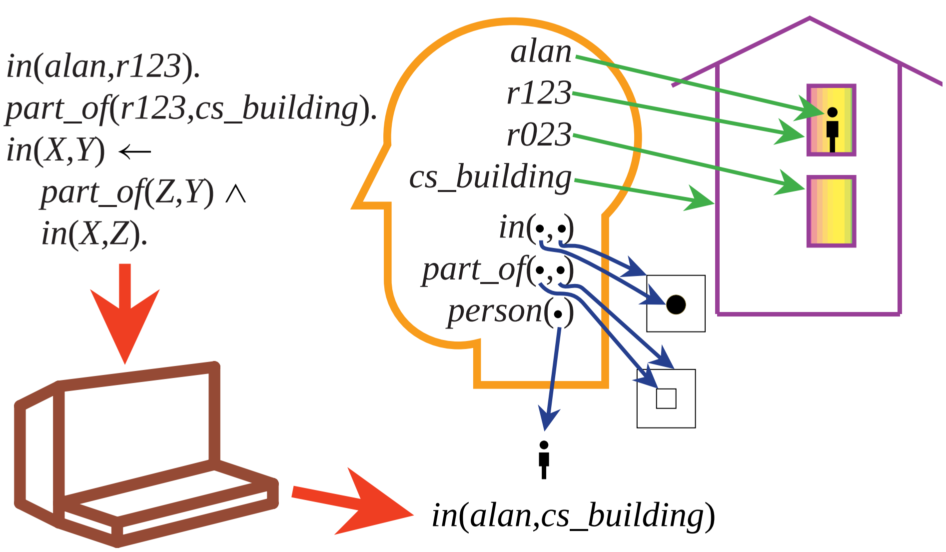
❷ Distinguish those things you want to talk about (the ontology).

❸ Choose symbols in the computer to denote objects and relations.

❹ Tell the system knowledge about the domain.

❺ Ask the system questions.

Role of Semantics in an RRS



Simplifying Assumptions of Initial 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.

$\Longrightarrow$ Datalog

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)

$\boxed{\text{definite clause}}$ is either an atomic symbol (a fact) or of the form:

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

where a and b_i are atomic symbols.

$\boxed{\text{query}}$ is of the form $?b_1 \wedge \cdots \wedge b_m$.

$\boxed{\text{knowledge base}}$ is a set of definite clauses.

Example Knowledge Base

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

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 & relations in world
  - constants denote individuals
  - predicate symbols denote relations

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.
- $\pi(p)$ 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

Each ground term denotes an individual in an interpretation.

A constant c $\boxed{\text{denotes in } I}$ the individual $\phi(c)$.

Ground (variable-free) atom $p(t_1, \ldots, t_n)$ is

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

Ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_m$ is $\boxed{\text{false in interpretation } I}$ if h is false in I and each b_i is true in I, and is $\boxed{\text{true in interpretation } I}$ 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 $\boxed{\text{model}}$ 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 $\boxed{\text{logical consequence}}$ of KB, written $\boxed{KB \models g,}$ 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.

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	$\pi(p)$	$\pi(q)$	$\pi(r)$	$\pi(s)$	
I_1	*TRUE*	*TRUE*	*TRUE*	*TRUE*	is a model of KB
I_2	*FALSE*	*FALSE*	*FALSE*	*FALSE*	not a model of KB
I_3	*TRUE*	*TRUE*	*FALSE*	*FALSE*	is a model of KB
I_4	*TRUE*	*TRUE*	*TRUE*	*FALSE*	is a model of KB
I_5	*TRUE*	*TRUE*	*FALSE*	*TRUE*	not a model of KB

$KB \models p$, $KB \models q$, $KB \not\models r$, $KB \not\models s$

User's view of Semantics

❶ Choose a task domain: intended interpretation.

❷ Associate constants with individuals you want to name.

❸ For each relation you want to represent, associate a predicate symbol in the language.

❹ Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.

❺ Ask questions about the intended interpretation.

❻ If $KB \models g$, 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 logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.

Variables

- Variables are $\boxed{\text{universally quantified}}$ in the scope of a clause.
- A $\boxed{\text{variable assignment}}$ 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 $\boxed{\text{for all}}$ variable assignments.

Queries and Answers

A $\boxed{\text{query}}$ is a way to ask if a body is a logical consequence of the knowledge base:

$$?b_1 \wedge \cdots \wedge b_m.$$

An $\boxed{\text{answer}}$ is either

- an instance of the query that is a logical consequence of the knowledge base *KB*, or
- $\boxed{\text{no}}$ if no instance is a logical consequence of *KB*.

Example Queries

$$KB = \begin{cases} in(alan, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	no
$?in(alan, r023).$	no
$?in(alan, B).$	$in(alan, r123)$
	$in(alan, cs\_building)$

Logical Consequence

Atom g is a logical consequence of KB if and only if:

- g is a fact in KB, or
- there is a rule

  $$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  in KB such that each b_i is a logical consequence of KB.

Debugging false conclusions

To debug answer g that is false in the intended interpretation:

- If g is a fact in KB, this fact is wrong.
- Otherwise, suppose g was proved using the rule:

  $$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  where each b_i is a logical consequence of KB.

  - If each b_i is true in the intended interpretation, this clause is false in the intended interpretation.
  - If some b_i is false in the intended interpretation, debug b_i.

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light

$light(l_1)$. $light(l_2)$.

% $down(S)$ is true if switch S is down

$down(s_1)$. $up(s_2)$. $up(s_3)$.

% $ok(D)$ is true if D is not broken

$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1). \Longrightarrow yes$

$?light(l_6). \Longrightarrow no$

$?up(X). \Longrightarrow up(s_2), up(s_3)$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \implies W = w_1$

$?connected\_to(w_1, W). \implies no$

$?connected\_to(Y, w_3). \implies Y = w_2, Y = w_4, Y = p_1$

$?connected\_to(X, W). \implies X = w_0, W = w_1, \ldots$

% $lit(L)$ is true if the light L is lit

$$lit(L) \leftarrow light(L) \wedge ok(L) \wedge live(L).$$

% $live(C)$ is true if there is power coming into C

$$live(Y) \leftarrow$$
$$connected\_to(Y, Z) \wedge$$
$$live(Z).$$
$$live(outside).$$

This is a $\boxed{\text{recursive definition}}$ of *live*.

Recursion and Mathematical Induction

$$above(X, Y) \leftarrow on(X, Y).$$
$$above(X, Y) \leftarrow on(X, Z) \wedge above(Z, Y).$$

This can be seen as:

- Recursive definition of *above*: prove *above* in terms of a base case (*on*) or a simpler instance of itself; or
- Way to prove *above* by mathematical induction: the base case is when there are no blocks between X and Y, and if you can prove *above* when there are n blocks between them, you can prove it when there are $n + 1$ blocks.

Limitations

Suppose you had a database using the relation:

$$enrolled(S, C)$$

which is true when student S is enrolled in course C.

You can't define the relation:

$$empty\_course(C)$$

which is true when course C has no students enrolled in it.

This is because $empty\_course(C)$ doesn't logically follow from a set of *enrolled* relations. There are always models where someone is enrolled in a course!