Interpretation and top-down and bottom-up ground proofs

Alex Muscar

Software Engineering Department
Faculty of Automation, Computers and Electronics
University of Craiova

March 7, 2011

Representation and Reasoning Systems

The use of knowledge in AI is done with a Representation and Reasoning System (*RRS*).

RRS = syntax + semantics + reasoning

RRS = formal language + precise semantics + proof procedure

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

Properties of an RRS

- The *proof procedure* uses a nondeterministic *search strategy*
- It offers two functions: *tell* and *ask*

Simplifying assumptions in a RRS

We can define the following simplifying assumptions for a RRS: *Individuals and Relations* (IR), *Definite Knowledge* (DK), *Static Environment* (SE) and *Finite Domain* (FD).

$$\text{RRS} + \text{IR} + \text{DK} + \text{SE} + \text{FD} \implies \textit{Datalog}$$

The syntax of Datalog

Variables a sequence of alphanumeric characters and '_', beginning with an upper-case letter

Constants a sequence of letters starting with a lower-case letter or a sequence of digits (a number)

Predicates a sequence of alphanumeric characters and '_', beginning with a lower-case letter

Terms variables or constants

Atoms are of the form p or $p(t_1, t_2, \ldots, t_n)$, where p is a predicate symbol and t_i are terms

The syntax of Datalog (cont'd)

Definite clauses are atomic symbols (facts) or have the form:

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

where a and b_i are atomic symbols.

Queries of the form $?b_1 \wedge \cdots \wedge b_n$

Knowledge base a set of definite clauses

Example knowledge base

Example

$$\texttt{depozitare(X, container\_metalic)} \leftarrow$$
$$\texttt{consistenta(X, solid)} \wedge$$
$$\texttt{proprietate(X, toxic).}$$
$$\texttt{depozitare(X, container\_metalic)} \leftarrow$$
$$\texttt{consistenta(X, solid)} \wedge$$
$$\texttt{proprietate(X, inflamabil).}$$
$$\texttt{depozitare(X, rezervor\_presurizat)} \leftarrow$$
$$\texttt{consistenta(X, gaz)} \wedge$$
$$\texttt{proprietate(X, inflamabil).}$$

Tarskian semantics

Semantics: specifies the meaning of sentences in a language. It specifies:

- a set of objects in the world (individuals); and
- a correspondence between the symbols in the language and objects (constants) and relations in the world (predicate symbols). interpretation function

Formal semantics

Interpretation: a triple $I = \langle D, \phi, \pi \rangle$, where:

- D is a nonempty set called the *domain* (D contains *real* objects);
- ϕ is a mapping that assigns to each constant c an element $\phi(c) \in D$; and
- π is a mapping that assigns to each n-ary predicate symbol p a function from D^n into $\{true, false\}$ (specifies wether the relation denoted by p is true or false for each n-tuple of individuals).

Example interpretation

- $D = \begin{array}{c}\{\text{the person named Alan,room 123,}\\ \text{room 023,the CS department's building}\}\end{array}.$
- The symbolic constants: $alan$, $r123$, $r023$, $cs\_building$.
- The ϕ function:
  - $\phi(alan) =$ the person named Alan
  - $\phi(r123) =$ room 123
  - $\phi(r023) =$ the room 023
  - $\phi(cs\_building) =$ the CS department's building
- The predicate symbols: $person/1$, $in/2$, $part\_of/2$.
- The π function:
  - $\pi(person) = \{(\text{the person named Alan})\}$
  - $\pi(in) = \begin{array}{c}\{(\text{the person named Alan,room 123}),\\ (\text{the person named Alan,the CS department's building})\}\end{array}$
  - $\pi(part\_of) = \begin{array}{l}\{(\text{the room 123,the CS department's building}),\\ (\text{the room 023,the CS department's building})\}\end{array}$

Truth in an interpretation

- The symbolic constant c denotes in I the individual $c^I = \phi(c) \in D$.
- The ground 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 otherwise.
- The ground clause $h \leftarrow b_1 \wedge \ldots \wedge b_n$ is false in interpretation I if h is false and every b_i is true, and true otherwise.

Models and logical consequences

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

Example

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

x	$\pi(p)$	$\pi(q)$	$\pi(r)$	$\pi(s)$
I_1	true	true	true	true
I_2	false	false	false	false
I_3	true	true	false	false
I_4	true	true	true	false
I_5	true	true	false	true

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

Proofs

Definition

A proof is a mechanically derivable demonstration that a formula derives from a knowledge base.

$\Delta \vdash g$ means that g can be derived from Δ
A proof can be sound ($\Delta \vdash g \Rightarrow \Delta \models g$) and complete ($\Delta \models g \Rightarrow \Delta \vdash g$)

Bottom-up and top-down proofs

Bottom-up proofs starting from the knowledge base and from the already proven facts new facts are obtained at every step.

Top-down proofs starting from the query each step works towards the knowledge base.

Bottom-up ground proofs

Only one rule of inference, a generalized form of *modus ponens*:

Definition

If $h \leftarrow b_1 \wedge \ldots \wedge b_n \text{in} \Delta$, and each b_i has been derived, then h can be derived.

This is using *forward chaining*.

The bottom-up proof procedure

$\Delta \vdash g$ if $g \in C$ at the end of this procedure:

Proof procedure

C := {};
repeat
    select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" $\in \Delta$:
        $\forall\ i,\ b_i \in C$, and
        $h \notin C$;
    $C := C \cup \{h\}$
until no more clauses can be selected.

Bottom-up proof example

Example

1. a ← b ∧ c.
2. b ← d ∧ e.
3. b ← g ∧ e.
4. c ← e.
5. d.
6. e.
7. f ← a ∧ g.

$$C := \emptyset$$
5: $C := \{d\}$
6: $C := \{d, e\}$
2: $C := \{d, e, b\}$
4: $C := \{d, e, b, c\}$
1: $C := \{a, d, e, b, c\}$
STOP

Top-down ground proofs

- Start from an *answer clause* of the form:
  $yes \leftarrow a_1 \wedge \ldots \wedge a_n$.
- Select an atom from the body of the answer clause, a_i, and a clause from Δ:
  $a_i \leftarrow b_1 \wedge \ldots \wedge b_m$.
- Using SLD resolution on the clause and a_i the new answer clause becomes:
  $yes \leftarrow a_1 \wedge \ldots \wedge a_{i-1} \wedge b_1 \wedge \ldots \wedge b_m \wedge a_{i+1} \wedge \ldots \wedge a_n$.

Proofs through SLD resolution

Resolution with a Linear selection function for Definite clauses (no disjunctions) – in Romanian: S - selectie; L - liniara; D - clauze precise

- A clause with $n = 0$ is an *answer* (i.e. $yes \leftarrow .$)
- A proof though SLD resolution for the $?q_1 \land \ldots \land q_k$ query from a knowledge base Δ is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that:
  - γ_0 is the initial answer clause: $yes \leftarrow q_1 \land \ldots \land q_k$,
  - γ_i is obtained by resolving γ_{i-1} with clauses in Δ, and
  - γ_n is an answer

The top-down proof procedure

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

Proof procedure

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

Nondeterminism

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

Top-down proof example (successful)

Example

1. a ← b ∧ c.
2. a ← e ∧ f.
3. b ← f ∧ k.
4. c ← e.
5. d ← k.
6. e.
7. f ← j ∧ e.
8. f ← c.
9. j ← c.

Query: $?a$

Selection: the leftmost atom from the answer clause body.

$$\begin{array}{lllll}
\gamma_0: & yes & \leftarrow & a & \\
\gamma_1(2): & yes & \leftarrow & e \wedge f & \\
\gamma_2(6): & yes & \leftarrow & f & \\
\gamma_3(8): & yes & \leftarrow & c & \\
\gamma_4(4): & yes & \leftarrow & e & \\
\gamma_5(6): & yes & \leftarrow & &
\end{array}$$

Top-down proof example (failing)

Example

1. a ← b ∧ c.
2. a ← e ∧ f.
3. b ← f ∧ k.
4. c ← e.
5. d ← k.
6. e.
7. f ← j ∧ e.
8. f ← c.
9. j ← c.

Query: $?a$

Selection: the leftmost atom from the answer clause body.

$$\begin{array}{llllllll}
\gamma_0: & yes & \leftarrow & a & & & & \\
\gamma_1(1): & yes & \leftarrow & b & \wedge & c & & \\
\gamma_2(3): & yes & \leftarrow & f & \wedge & k & \wedge & c \\
\gamma_3(8): & yes & \leftarrow & c & \wedge & k & \wedge & c \\
\gamma_4(4): & yes & \leftarrow & e & \wedge & k & \wedge & c \\
\gamma_5(6): & yes & \leftarrow & k & \wedge & c & &
\end{array}$$