Proposições e Inferência

22 de janeiro de 2024

Source

Chapter 5 - `https://artint.info/index.html`

1 Propositional Definite Clauses

The language of propositional definite clauses is a subset of propositional calculus that does not allow uncertainty or ambiguity. In this language, propositions have the same meaning as in propositional calculus, but not all compound propositions are allowed in a knowledge base.

The syntax of propositional definite clauses is defined as follows:

- An atomic proposition or atom is the same as in propositional calculus.
- A definite clause is of the form

$$h \leftarrow a_1 \wedge \ldots \wedge a_n.$$

where h is an atom, the head of the clause, and each a_i is an atom. It can be read "h if a_1 and ... and a_n". If $n > 0$, the clause is called a rule, where $a_1 \wedge \ldots \wedge a_n$ is the body of the clause.

- If $n = 0$, the arrow can be omitted and the clause is called an atomic clause or fact. The clause has an empty body.

A knowledge base is a set of definite clauses.

2 Queries and Answers

1. One reason to build a description of a real or imaginary world is to be able to determine what else must be true in that world.
2. After the computer is given a knowledge base about a particular domain, a user might like to ask the computer questions about that domain.
3. The computer can answer whether or not a proposition is a logical consequence of the knowledge base. A user that knows the meaning of the atoms, for example when is $p \wedge q \wedge r_3$ true, can interpret the answer in terms of the domain.
4. A query is a way of asking whether a proposition is a logical consequence of a knowledge base. Once the system has been provided with a knowledge base, a query is used to ask whether a proposition is a logical consequence of the knowledge base. Queries have the form

$$\text{ask} \quad p.$$

where p is an atom or a conjunction of atoms (analogous to the body of a rule).

5. A query is a question that has the answer yes if the body is a logical consequence of the knowledge base, or the answer no if the body is not a consequence of the knowledge base. The latter does not mean that $p \wedge q \wedge r$ is false in the intended interpretation but rather that it is impossible to determine whether it is true or false based on the knowledge provided.

3 Proofs

1. So far, we have specified what an answer is, but not how it can be computed. The definition of $\models$ specifies which propositions should be logical consequences of a knowledge base but not how to compute them.

2. The problem of deduction is to determine if some proposition is a logical consequence of a knowledge base. Deduction is a specific form of inference.

3. A proof is a mechanically derivable demonstration that a proposition logically follows from a knowledge base.

4. A theorem is a provable proposition. A proof procedure is a – possibly non-deterministic – algorithm for deriving consequences of a knowledge base.

5. Given a proof procedure, $KB \vdash p$ means p can be proved or derived from knowledge base KB.

6. A proof procedure's quality can be judged by whether it computes what it is meant to compute.

7. A proof procedure is **sound** with respect to the semantics if everything that can be derived from a knowledge base is a logical consequence of the knowledge base. That is, if $KB \vdash p$, then $KB \models p$.

8. A proof procedure is **complete** with respect to the semantics if there is a proof of each logical consequence of the knowledge base. That is, if $KB \models p$, then $KB \vdash p$.

3.1 Botton-up

Algorithm 1 Bottom-up proof procedure for computing consequences of KB

procedure Prove_DC_BU(KB)
Inputs
  KB: a set of definite clauses
Output
  Set of all atoms that are logical consequences of KB
Local
  C is a set of atoms
  $C := \{\}$
repeat
  select "$h \leftarrow p_1 \wedge \ldots \wedge p_n$" in KB where $p_i \in C$ for all i, and $h \notin C$
  $C := C \cup \{h\}$
until no more definite clauses can be selected
return C

3.2 Top-down

Algorithm 2 Top-down definite clause proof procedure

non-deterministic procedure Prove_DC_TD(KB, Q)
Inputs
  KB: a set of definite clauses
  Q: a set of atoms to prove
Output
  YES if $KB \models Q$ and the procedure fails otherwise
Local
  C is a set of atoms
  $C := Q$
repeat
  select an atom p in C
  choose definite clause "$p \leftarrow B$" in KB with p as head
  $C := C \cup (B \setminus \{p\})$
until $C = \{\}$
return YES