Inference And Planning

Arnav Gupta

October 18, 2024

Contents

1 Problem Solving **1**
1.1 Logical Consequence 2

2 Proofs **2**
2.1 Bottom-Up Proof 3
2.2 Top-Down Proof 3

3 Planning and Actions **4**
3.1 Forward Planning 5
3.2 Regression Planning 5

1 Problem Solving

Two methods for solving problems:

- **procedural**
  - devise an algorithm
  - program an algorithm
  - execute the program
- **declarative**
  - identify knowledge needed
  - encode the knowledge in a representation (knowledge base - KB)
  - use logical consequences of KB to solve the problem

A logic consists of **syntax**, **semantics**, and **proof procedure**.

Proof: a sequence of sentences derivable using an inference rule

1.1 Logical Consequence

For a set of **statements** $\{X\}$:

- a set of truth assignments is an **interpretation**
- an interpretation that makes it true is a **model**
- if it has no model, it is **inconsistent**

A statement is a **logical consequence** of a set of statements if the statement is true in every model of the set.

P	H	C	$P \to (\neg H \to C)$	$P \to \neg H$	$P \to C$
F	F	F	T	T	T
F	F	T	T	T	T
F	T	F	T	T	T
F	T	T	T	T	T
T	F	F	T	F	F
T	T	F	T	F	T
T	T	T	T	F	T
T	F	T	T	T	T

Table 1: Truth table with highlighted models, showing $P_1, P_2 \models D$.

An argument is **valid** if:

- conclusions are logical consequences of the premise
- conclusions are true in every model of the premises

2 Proofs

Knowledge Base: set of axioms

KB $\vdash g$ means g can be derived from KB using the proof procedure. If it is true, then g is a theorem.

Soundness: if KB $\vdash g$, then KB $\models g$.

Completeness: if KB $\models g$, then KB $\vdash g$.

Assume a **closed world**:

- agent knows everything
- if it cannot prove something, it must be false
- negation is failure

2.1 Bottom-Up Proof

Forward chaining: start from facts and use rules to generate all possible atoms

Algorithm 1 Bottom-Up Proof

```
C ← {}                                  ▷ Initialize the set of conclusions
repeat
    Select r ∈ KB such that
        r is h ← b_1 ∧ ... ∧ b_m
        b_i ∈ C ∀i                      ▷ All premises are in C
        h ∉ C                           ▷ h is not already in C
    C ← C ∪ {h}                         ▷ Add h to the conclusions
until no more clauses can be selected
```

Forward chaining is sound and complete.

2.2 Top-Down Proof

Start with query and work backwards, trying to see if it is proved from the premises.

Algorithm 2 Top-Down Proof

```
procedure SOLVE(q_1 ∧ ... ∧ q_k)
    ac ← "yes ← q_1 ∧ ... ∧ q_k"        ▷ Initialize the answer condition
    repeat
        Select a conjunct q_i from the body of ac
        Choose a clause C from KB with q_i as the head
        Replace q_i in the body of ac by the body of C
    until ac is an answer
end procedure
```

In the select step, some selections will lead to solutions more quickly, though any should lead to a solution in the end. In the choose step, if one choice doesn't give a solution, others may, so all must be tried.

KB can contain **relations** (predicates) or **quantification**.

3 Planning and Actions

Planning: decide what to do based on the agent's ability, goals, and state of the world; basically find a sequence of actions to the goal

Assume:

- a single agent
- deterministic world
- no events outside the agent's control that change the state of the world
- agent knows what state the world is in (full observability)
- time progresses discreetly
- goals are predicates of states that must be achieved/maintained

Action: partial function (some actions only possible from some states) from states to states

Preconditions of an action specify if it can occur. **Effect** of an action specifies resulting state.

Causal rules specify when a feature gets a new value. **Frame rules** specify when the feature keeps its values.

In planning, the givens are:

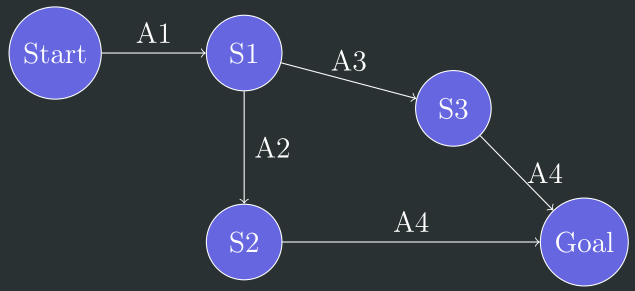
- description of effects and preconditions of actions
- description of initial state
- goal to achieve

Achieved by finding a sequence of possible actions that will result is state that satisfies the goal.

3.1 Forward Planning

Search in the state-space graph:

- nodes represent states
- arcs correspond to actions legal from that state
- plan is a path from the initial state to a goal state
- heuristics can be important



3.2 Regression Planning

Search backwards from the goal description with nodes corresponding to subgoals and arcs to actions:

- nodes are propositions (assignments of values to features)
- arcs correspond to actions that can achieve goals
- neighbours of a node specify what must be true immediately before the arc so that the node is true immediately after
- start node is the goal to be achieved
- goal(N) is true if N is a proposition true of the initial state