STATE SPACE SEARCH - A state space is represented by a four-tuple [N,A,S,GD], where:

N - is the set of nodes or states of the graph. These correspond to the states in a problem-solving process.
A - is the set of arcs (or links) between nodes. These correspond to the steps in a problem-solving process.
S - a nonempty subset of N, contains the start state(s) of the problem.
GD - a nonempty subset of N, contains the goal state(s) of the problem. The states in GD are described using either:

1. A measurable property of the states encountered in the search.
2. A measurable property of the path developed in the search, for example, the sum of the transition costs for the arcs of the path.

A solution path is a path through this graph from a node in S to a node in GD.

Logic as State Space Language

- **Propositional and predicate logic** can represent:
  - Nodes: logical **assertions** or **truth values**
  - Arcs: logical **inference rules**
- **Search algorithms** can then traverse this graph to reach conclusions (e.g., derive a goal formula from axioms).

EXAMPLE 3.3.1: THE PROPOSITIONAL CALCULUS

The first example of how a set of logic relationships may be viewed as defining a graph is from the propositional calculus. If p, q, r,... are propositions, assume the assertions:

$q \rightarrow p$
$r \rightarrow p$
$v \rightarrow q$
$s \rightarrow r$
$t \rightarrow r$
$s \rightarrow u$
s
t

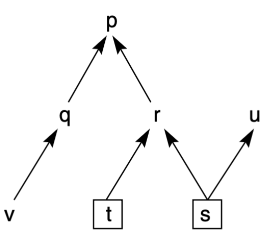


Figure 3.20 State space graph of a set of implications in the propositional calculus.

In Figure 3.20 the arcs correspond to logical implications (→). Propositions that are given as true (s and t) correspond to the given data of the problem

FYI - **Forward-Chaining** (start from known facts and infer new facts) and **Backward-Chaining** (start from goal and trace backward)

EXAMPLE 3.3.2: AND/OR GRAPH SEARCH

The second example is also from the propositional calculus but generates a graph that contains both and and or descendants. Assume a situation where the following propositions are true:

a
b
c
$a \land b \rightarrow d$
$a \land c \rightarrow e$
$b \land d \rightarrow f$
$f \rightarrow g$
$a \land e \rightarrow h$

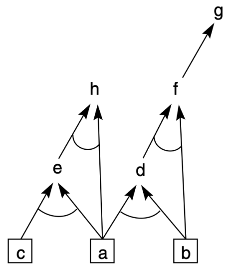


Figure 3.23 And/or graph of a set of propositional calculus expressions.

This set of assertions generates the and/or graph in Figure 3.23.

Questions that might be asked (answers deduced by the search of this graph) are:

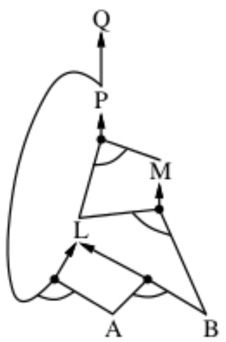
1. Is h true? A and C are true -> E is true -> A and E are true therefore h is true.

2. Is h true if b is no longer true?

More examples:

$P \Rightarrow Q$
$L \land M \Rightarrow P$
$B \land L \Rightarrow M$
$A \land P \Rightarrow L$
$A \land B \Rightarrow L$
A
B

(a)



(b)

Figure 7.15 (a) A simple knowledge base of Horn clauses. (b) The corresponding AND–OR graph.

Consider the following propositional logic KB in Horn form:

- A1: $A \wedge B \implies D$
- A2: $A \wedge F \implies G$
- A3: $B \wedge C \implies F$
- A4: $D \wedge C \implies H$
- A5: $A \wedge C \implies E$
- A6: $G \wedge E \wedge H \implies I$
- A7: A
- A8: B
- A9: C

---

4. (30 pt) Consider the following knowledge base
   a. Prove that Q is true with:
      1. $P \rightarrow Q$
      2. $L \wedge M \rightarrow P$
      3. $B \wedge L \rightarrow M$
      4. $A \wedge P \rightarrow L$
      5. $A \wedge B \rightarrow L$
      6. A
      7. B
      i. (4 pt) Forward-Chaining
      ii. (4 pt) Backward-Chaining

i. Forward-Chaining (start from known facts and infer new facts)

Known:

- A
- B

Step 1:
From (5): A ∧ B → L → infer **L**

Step 2:
From (3): B ∧ L → M → infer **M**

Step 3:
From (2): L ∧ M → P → infer **P**

Step 4:
From (1): P → Q → infer **Q**

Q is inferred by forward chaining

Backward-Chaining (start from Q and trace backward)

Goal: Q

- To prove Q, need P (from 1: P → Q)
- To get P, need L ∧ M (from 2)
  - To get M, need B ∧ L (from 3)
  - We already have B
  - To get L:
    - (5): A ∧ B → L (we have both A and B) → infer **L**
  - Now have B and L → infer **M**
- Now L and M → infer **P**
- P → Q → infer **Q**

Q is proven by backward chaining