Searching

➤ Often we are not given an algorithm to solve a problem, but only a specification of what is a solution — we have to search for a solution.

➤ Search is a way to implement don’t know nondeterminism.

➤ So far we have seen how to convert a semantic problem of finding logical consequence to a search problem of finding derivations.
Search Graphs

➤ A **graph** consists of a set \( N \) of **nodes** and a set \( A \) of ordered pairs of nodes, called **arcs**.

➤ Node \( n_2 \) is a **neighbor** of \( n_1 \) if there is an arc from \( n_1 \) to \( n_2 \). That is, if \( \langle n_1, n_2 \rangle \in A \).

➤ A **path** is a sequence of nodes \( \langle n_0, n_1, \ldots, n_k \rangle \) such that \( \langle n_{i-1}, n_i \rangle \in A \).

➤ Given a set of **start nodes** and **goal nodes**, a **solution** is a path from a start node to a goal node.
Example Graph for the Delivery Robot

- r123
- o125 → o123 → o119
- l3d1 → l3d2 → l3d3
- l2d1 → l2d2 → l2d3 → l2d4
- mail → ts → o103 → o109 → o111
- storage
Search Graph for SLD Resolution

\[ a \leftarrow b \land c \]
\[ a \leftarrow g \]
\[ a \leftarrow h \]
\[ b \leftarrow j \]
\[ b \leftarrow k \]
\[ d \leftarrow m \]
\[ d \leftarrow p \]
\[ f \leftarrow m \]
\[ f \leftarrow p \]
\[ g \leftarrow m \]
\[ g \leftarrow f \]
\[ k \leftarrow m \]
\[ h \leftarrow m \]
\[ p \]
\[ \text{yes} \leftarrow a \land d \]
\[ \text{yes} \leftarrow b \land c \land d \]
\[ \text{yes} \leftarrow g \land d \]
\[ \text{yes} \leftarrow h \land d \]
\[ \text{yes} \leftarrow m \land d \]
\[ \text{yes} \leftarrow m \land d \]
\[ \text{yes} \leftarrow f \land d \]
\[ \text{yes} \leftarrow k \land c \land d \]
\[ \text{yes} \leftarrow p \land d \]
\[ \text{yes} \leftarrow m \land c \land d \]
\[ \text{yes} \leftarrow m \land c \land d \]
\[ \text{yes} \leftarrow m \]
\[ \text{yes} \leftarrow p \]

\[ \text{yes} \]
Graph Searching

➤ Generic search algorithm: given a graph, start nodes, and goal nodes, incrementally explore paths from the start nodes.

➤ Maintain a **frontier** of paths from the start node that have been explored.

➤ As search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered.

➤ The way in which the frontier is expanded defines the **search strategy**.
Problem Solving by Graph Searching
Graph Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure \textit{goal}(n) that tests if \( n \) is a goal node.

\( \text{frontier} := \{ \langle s \rangle : s \text{ is a start node} \} \);

\textbf{while} \text{frontier} is not empty:

\textbf{select} and \textbf{remove} path \( \langle n_0, \ldots, n_k \rangle \) from \text{frontier};

\textbf{if} \( \text{goal}(n_k) \)

\textbf{return} \( \langle n_0, \ldots, n_k \rangle \);

\textbf{for every} neighbor \( n \) of \( n_k \)

\textbf{add} \( \langle n_0, \ldots, n_k, n \rangle \) to \text{frontier};

\textbf{end while}
We assume that after the search algorithm returns an answer, it can be asked for more answers and the procedure continues.

Which value is selected from the frontier at each stage defines the search strategy.

The neighbors defines the graph.

is_goal defines what is a solution.