Propositional logic program

A *definite clause* is a formula of one of the two shapes below

$$q \qquad \text{(a Prolog } \textit{fact}\ q\,.\ \text{)}$$

$$p_1 \wedge \cdots \wedge p_k \ \rightarrow q \qquad \text{(a Prolog } \textit{rule}\ q\ \texttt{:-}\ p_1, \ldots, p_k.\text{)}$$

where $p_1, \ldots, p_k, q$ are all *atoms* (that is, atomic statements).

A *logic program* is a list $F_1, \ldots, F_n$ of definite clauses

A *goal* is a list $g_1, \ldots, g_m$ of atoms.

The job of the system is to ascertain whether the logical consequence below holds.

$$F_1, \ldots, F_n \models g_1 \wedge \cdots \wedge g_m \ .$$

Inference system

The definite clauses $F_1, \ldots, F_n$ in the program are taken as axioms.

We use a single inference rule to derive new *established goals* from goals already established

$$\frac{g_1, \ldots, g_{l-1}, h_1, \ldots, h_k, g_{l+1}, \ldots, g_m \qquad h_1 \wedge \cdots \wedge h_k \rightarrow g_l}{g_1, \ldots, g_{l-1}, g_l, g_{l+1}, \ldots, g_m}$$

where $1 \leq l \leq m$ and $k \geq 0$.

A *derivation* (or *proof*) of a goal $g_1, \ldots, g_m$ is a tree of applications of the rule, in which every leaf is one of the definite clauses $F_1, \ldots, F_n$ comprising the program, and the root of the tree is the goal $g_1, \ldots, g_m$.

Example logic program

$$
\begin{aligned}
\texttt{hotterSun} &\rightarrow \texttt{warmerClimate} \\
\texttt{carbonIncrease} &\rightarrow \texttt{warmerClimate} \\
\texttt{warmerClimate} &\rightarrow \texttt{iceMelts} \\
\texttt{iceMelts} &\rightarrow \texttt{albedoDecrease} \\
\texttt{albedoDecrease} &\rightarrow \texttt{warmerClimate} \\
&\quad\ \texttt{carbonIncrease}
\end{aligned}
$$

A derivation of the goal `iceMelts`

$$\dfrac{\dfrac{\texttt{carbonIncrease} \quad \texttt{carbonIncrease} \longrightarrow \texttt{warmerClimate}}{\texttt{warmerClimate}} \qquad \texttt{warmerClimate} \longrightarrow \texttt{iceMelts}}{\texttt{iceMelts}}$$

Top-down proof search

A top-down search for a derivation maintains a list $g_1, \ldots, g_n$ of atoms, the *current goal*. The procedure is:

- For l with $1 \leq l \leq n$, find an axiom (program clause) of form

$$h_1 \wedge \cdots \wedge h_k \;\rightarrow\; g_l$$

  Replace g_l with $h_1, \ldots, h_k$. So the current goal becomes

$$g_1, \ldots, g_{l-1}, h_1, \ldots, h_k, g_{l+1}, \ldots, g_n$$

  This includes, with $k = 0$, the case in which g_l is itself an axiom, in which case it simply gets removed from the list.

The choice of l and choice of axiom means that the search space branches, producing a *search tree*.

The goal of the proof search procedure is to find a branch in the search tree ending in a leaf labelled with the empty list [].

Prolog search tree

Prolog search specializes the above by always considering the individual goal atoms in *goal order*.

This means that given a current goal

$$g_1, \ldots, g_n$$

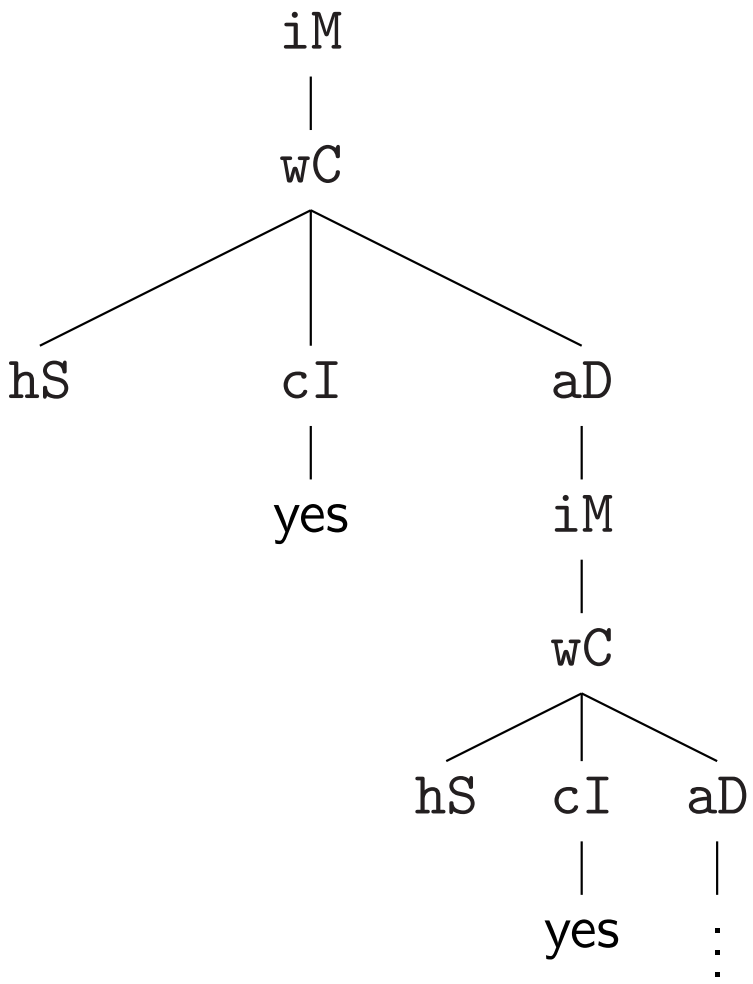
Prolog always looks for an axiom (program clause) of form

$$h_1 \wedge \cdots \wedge h_k \;\rightarrow\; g_1$$

That is l on the previous slide is always chosen to be 1.

This policy is built into the Prolog search tree for a query. In a Prolog search tree, the branching reflects only the choice involved in selecting which axiom to use.

Prolog search tree for goal `iceMelts`



Prolog proof search

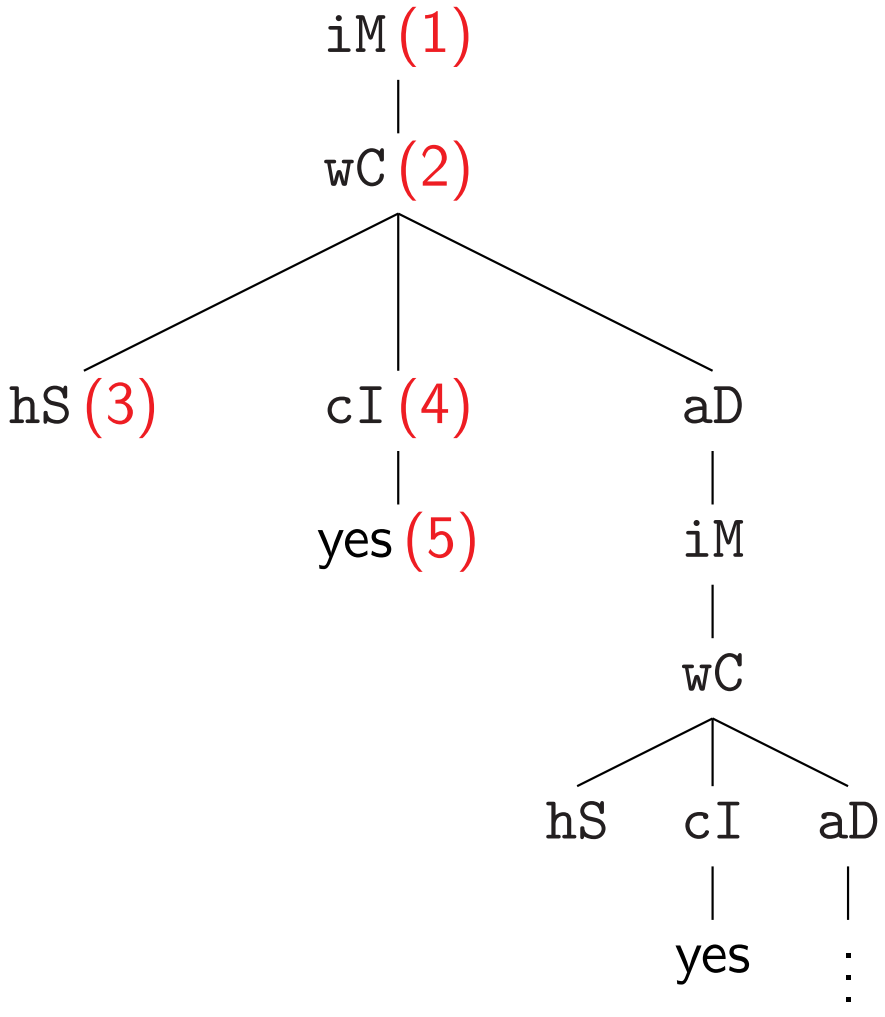
Prolog proof search is *depth first*:

- The search always moves to an *unvisited* (i.e., not previously visited) *child* (i.e., immediately below) node of the current node, whenever such a node exists.
- If there is no such node, the search backtracks to the most recently visited node from which such an unvisited child node is available.

Prolog proof search follows *program order*.

- Child nodes are visited in the order that the axioms (Prolog rules) that determine the child node appear in the program (i.e., from left to right in the trees as we are drawing them).

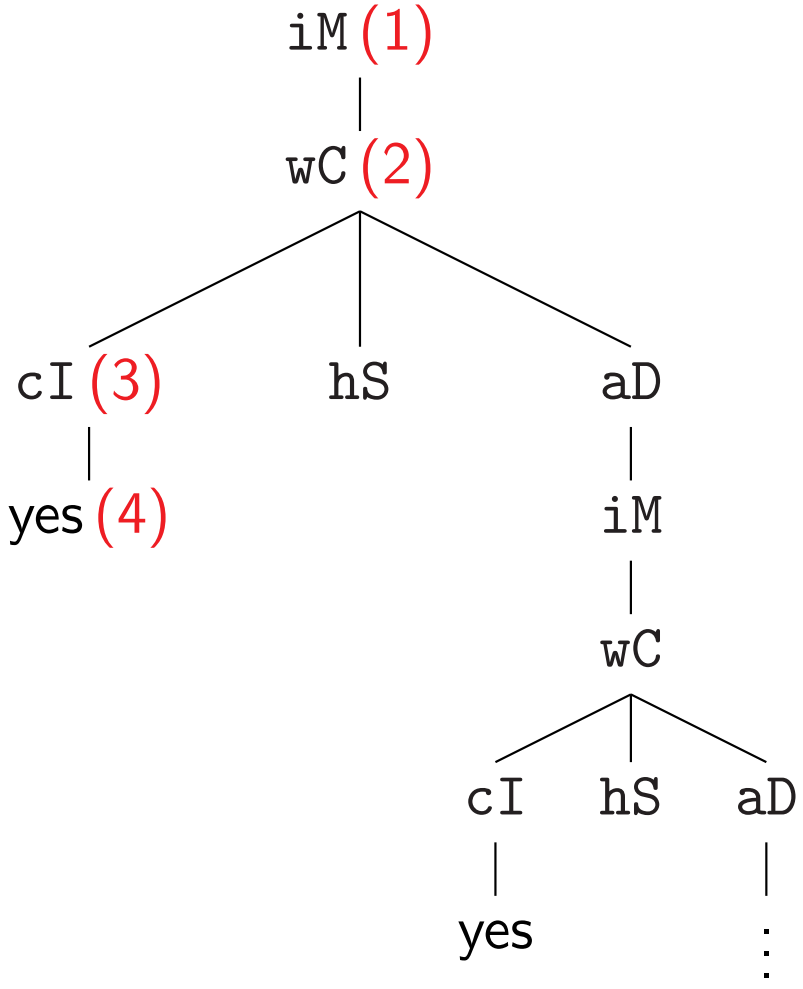
Prolog search for goal `iceMelts`



Reordering program can improve efficiency . . .

```
carbonIncrease   →  warmerClimate
    hotterSun    →  warmerClimate
warmerClimate    →  iceMelts
     iceMelts    →  albedoDecrease
albedoDecrease   →  warmerClimate
                    carbonIncrease
```

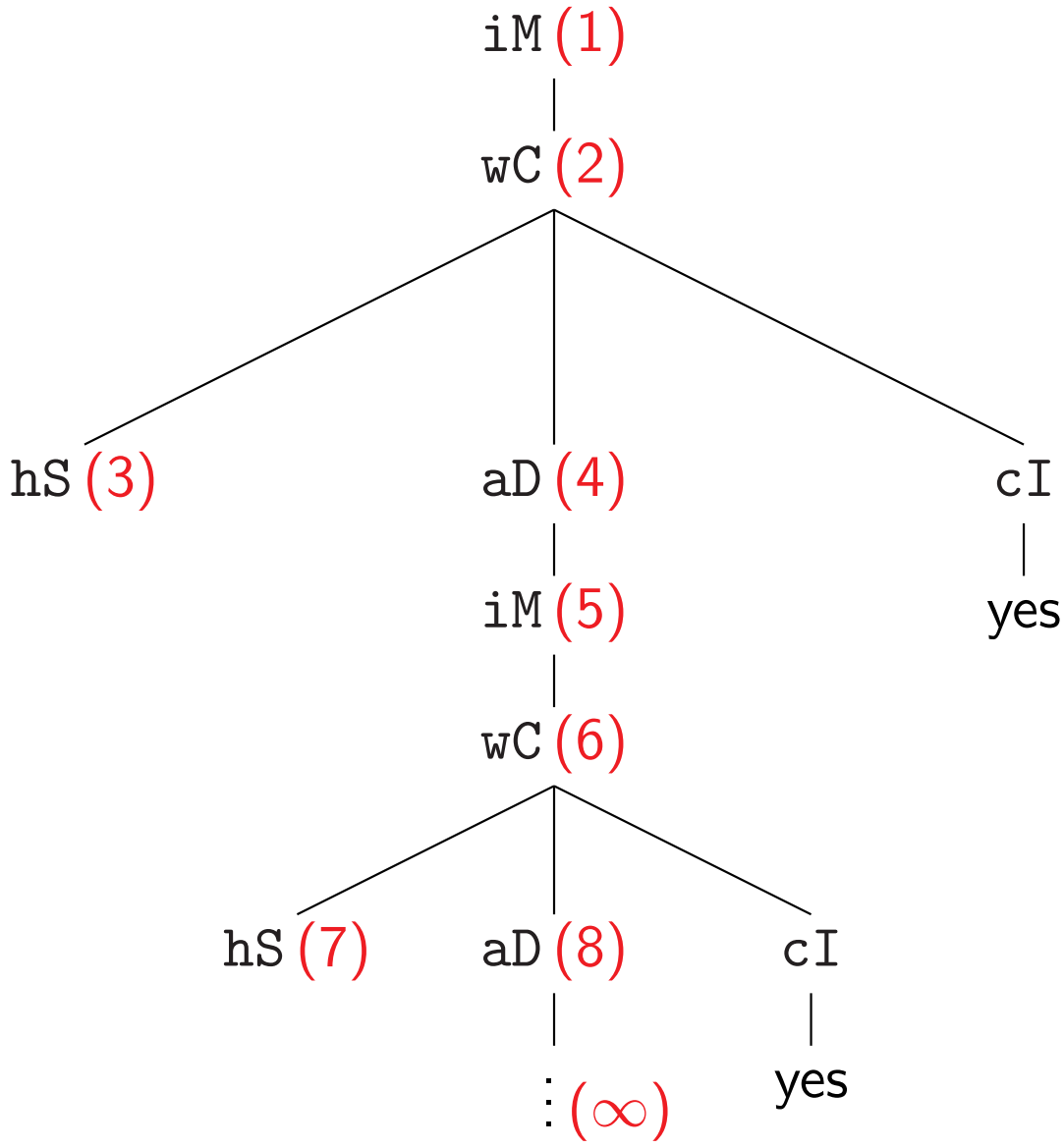
Resulting Prolog search



. . . or reduce efficiency . . .

$$
\begin{aligned}
\texttt{hotterSun} &\rightarrow \texttt{warmerClimate} \\
\texttt{albedoDecrease} &\rightarrow \texttt{warmerClimate} \\
\texttt{carbonIncrease} &\rightarrow \texttt{warmerClimate} \\
\texttt{warmerClimate} &\rightarrow \texttt{iceMelts} \\
\texttt{iceMelts} &\rightarrow \texttt{albedoDecrease} \\
&\quad\ \texttt{carbonIncrease}
\end{aligned}
$$

. . . drastically!



In Sicstus Prolog

Program:

```
warmerClimate :- hotterSun.
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.
hotterSun :- false.
```

Query:

```
| ?- iceMelts.
!  Resource error:  insufficient memory
```

Incompleteness of Prolog proof search

The correct answer to the query `iceMelts` is `yes`, because `iceMelts` is a logical consequence of the program (irrespective of how we order the axioms in the program).

Our original derivation of `iceMelts` is a valid derivation (irrespective of how we order the axioms in the program).

However, whether Prolog proof search is successful or not depends on the order in which the axioms in the program are given.

The Prolog proof search procedure is said to be *incomplete*: it does not always find a derivation, even if a derivation exists.

Notation

- Goal $g_1, \ldots, g_m$ is a *logical consequence* of $F_1, F_2, \ldots, F_n$

$$F_1, \ldots, F_n \models g_1 \wedge \cdots \wedge g_m \ .$$

- Goal $g_1, \ldots, g_m$ is *derivable* from $F_1, F_2, \ldots, F_n$:

$$F_1, \ldots, F_n \vdash g_1, \ldots, g_m \ .$$

  i.e., there is some derivation of $g_1, \ldots, g_m$ from axioms $F_1, F_2, \ldots, F_n$

- Goal $g_1, \ldots, g_m$ is *Prolog derivable* from $F_1, F_2, \ldots, F_n$:

$$F_1, \ldots, F_n \vdash_{\mathsf{Prolog}} g_1, \ldots, g_m \ ,$$

  i.e., Prolog proof search is successful.

Fundamental properties

Correctness of Prolog proof search

$$F_1, \ldots, F_n \vdash_{\mathsf{Prolog}} g_1, \ldots, g_m \quad \text{implies} \quad F_1, \ldots, F_n \vdash g_1, \ldots, g_m \ .$$

Soundness of inference system

$$F_1, \ldots, F_n \vdash g_1, \ldots, g_m \quad \text{implies} \quad F_1, \ldots, F_n \models g_1 \wedge \cdots \wedge g_m \ .$$

Completeness of inference system

$$F_1, \ldots, F_n \models g_1 \wedge \cdots \wedge g_m \quad \text{implies} \quad F_1, \ldots, F_n \vdash g_1, \ldots, g_m \ .$$

Incompleteness of Prolog proof search

$$F_1, \ldots, F_n \models g_1 \wedge \cdots \wedge g_m \text{ doesn't imply } F_1, \ldots, F_n \vdash_{\mathsf{Prolog}} g_1, \ldots, g_m \ .$$