% $\text{aprove}(G)$ is true if $G$ is a logical consequence of the base-level KB and yes/no answers provided by the user.

$\text{aprove}(\text{true})$.

$\text{aprove}((A \& B)) \leftarrow \text{aprove}(A) \land \text{aprove}(B)$.

$\text{aprove}(H) \leftarrow \text{askable}(H) \land \text{answered}(H, \text{yes})$.

$\text{aprove}(H) \leftarrow$

$\text{askable}(H) \land \text{unanswered}(H) \land \text{ask}(H, \text{Ans}) \land$

$\text{record}(\text{answered}(H, \text{Ans})) \land \text{Ans} = \text{yes}$.

$\text{aprove}(H) \leftarrow (H \Leftarrow B) \land \text{aprove}(B)$. 
% wprove(G, A) is true if G follows from base-level KB, and A is a list of ancestor rules for G.

wprove(true, Anc).

wprove((A & B), Anc) ←
  wprove(A, Anc) ∧
  wprove(B, Anc).

wprove(H, Anc) ←
  (H ⇐ B) ∧
  wprove(B, [(H ⇐ B)|Anc]).
Delaying Goals

Some goals, rather than being proved, can be collected in a list.

- To delay subgoals with variables, in the hope that subsequent calls will ground the variables.
- To delay assumptions, so that you can collect assumptions that are needed to prove a goal.
- To create new rules that leave out intermediate steps.
- To reduce a set of goals to primitive predicates.
% $dprove(G, D_0, D_1)$ is true if $D_0$ is an ending of list of delayable atoms $D_1$ and $KB \land (D_1 \setminus D_0) \models G$.

dprove(true, D, D).

dprove((A \& B), D_1, D_3) \leftarrow

\quad dprove(A, D_1, D_2) \land dprove(B, D_2, D_3).

dprove(G, D, [G|D]) \leftarrow delay(G).

dprove(H, D_1, D_2) \leftarrow

\quad (H \leftarrow B) \land dprove(B, D_1, D_2).
live(W) ⇐
    connected_to(W, W₁) &
    live(W₁).
live(outside) ⇐ true.
connected_to(w₆, w₅) ⇐ ok(cb₂).
connected_to(w₅, outside) ⇐ ok(outside_connection).
delay(ok(X)).
?dprove(live(w₆), [], D).
% $hprove(G, T)$ is true if $G$ can be proved from the base-level KB, with proof tree $T$.

$hprove(true, true)$.

$hprove((A \& B), (L \& R)) \leftarrow$

$hprove(A, L) \land$

$hprove(B, R)$.

$hprove(H, if(H, T)) \leftarrow$

$(H \leftarrow B) \land$

$hprove(B, T)$.