# Ask-the-user meta-interpreter

% $aprove(G)$ is true if $G$ is a logical consequence of the

% base-level KB and yes/no answers provided by the user.

$$aprove(true).$$

$$aprove((A \,\&\, B)) \leftarrow aprove(A) \wedge aprove(B).$$

$$aprove(H) \leftarrow askable(H) \wedge answered(H, yes).$$

$$aprove(H) \leftarrow$$

$$askable(H) \wedge unanswered(H) \wedge ask(H, Ans) \wedge$$

$$record(answered(H, Ans)) \wedge Ans = yes.$$

$$aprove(H) \leftarrow (H \Leftarrow B) \wedge aprove(B).$$

# Meta-interpreter to collect rules for WHY

% $wprove(G, A)$ is true if $G$ follows from base-level KB, and

% $A$ is a list of ancestor rules for $G$.

$$wprove(true, Anc).$$

$$wprove((A \,\&\, B), Anc) \leftarrow$$

$$\quad wprove(A, Anc) \wedge$$

$$\quad wprove(B, Anc).$$

$$wprove(H, Anc) \leftarrow$$

$$\quad (H \Leftarrow B) \wedge$$

$$\quad wprove(B, [(H \Leftarrow B)|Anc]).$$

# 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.

# Delaying Meta-interpreter

% $dprove(G, D_0, D_1)$ is true if $D_0$ is an ending of list of

% delayable atoms $D_1$ and $KB \wedge (D_1 - D_0) \models G$.

$dprove(true, D, D).$

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

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

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

$dprove(H, D_1, D_2) \leftarrow$

$\quad (H \Leftarrow B) \wedge dprove(B, D_1, D_2).$

# Example base-level KB

$live(W) \Leftarrow$

$\quad connected\_to(W, W_1)$ &

$\quad live(W_1)$.

$live(outside) \Leftarrow true$.

$connected\_to(w_6, w_5) \Leftarrow ok(cb_2)$.

$connected\_to(w_5, outside) \Leftarrow ok(outside\_connection)$.

$delay(ok(X))$.

$?dprove(live(w_6), [\,], D)$.

# Trace of dprove example

Each forward step is indicated as a box:

?<goal of proof step> <parent box#> <box#>
<matching clause of knowledge base before unification>
<matching clause of knowledge base after unification>

The box colors indicate:

fail

success

subgoal calls

Subgoal successes are fed back to the parent box:

<proved goal> <box#> -> <parent box#>

# Proof steps using dprove (1)

?dprove(live(w6), [ ], D) 0 1
dprove(G, D, [G| D]) <- delay(G)
dprove(live(w6), [ ], [live(w(6)]) <- delay(live(w6))

?dprove(live(w6), [ ], D) 1 2
dprove(H, D1, D2) <- (H <= B) ∧dprove(B, D1, D2)
dprove(live(w6), [ ], D) <- (live(w6) <= B) ∧dprove(B, [ ], D)

?(live(w6) <= B) 2 3
live(W) <= connected_to(W, W1) & live(W1)
live(w6) <= connected_to(w6, W1) & live(W1)

?dprove(connected_to(w6, W1) & live(W1), [ ], D) 2 4
dprove((A & B), D4, D6) <- dprove(A, D4, D5) ∧dprove(B, D5, D6)
dprove((connected_to(w6, W1) & live(W1)), [ ], D) <-
dprove(connected_to(w6, W1), [ ], D5) ∧dprove(live(W1), D5, D)

# Proof steps using dprove (2)

?dprove(connected_to(w6, W1), [ ], D5) 4 5
dprove(G, D, [G| D]) <- delay(G)
dprove(connected_to(w6, W1), [ ], [connected_to(w6, W1)]) <-
delay(connected_to(w6, W1))

?dprove(connected_to(w6, W1), [ ], D5) 4 6
dprove(H, D7, D8) <- (H <= B) ∧dprove(B, D7, D8)
dprove(connected_to(w6, W1), [ ], D5) <- (connected_to(w6, W1) <= B)
∧dprove(B, [ ], D5)

?(connected_to(w6, W1) <= B) 6 7
connected_to(w6, w5) <= ok(cb2)
connected_to(w6, w5) <= ok(cb2)

?dprove(ok(cb2), [ ], D5) 6 8
dprove(G, D9, [G| D9]) <- delay(G)
dprove(ok(cb2), [ ], [ok(cb2)]) <- delay(ok(cb2))

# Proof steps using dprove (3)

?delay(ok(cb2)) 8 9
delay(ok(X))
delay(ok(cb2))
dprove(ok(cb2), [ ], [ok(cb2)]) <- true
dprove(connected_to(w6, w5), [ ], [ok(cb2)]) <- true

dprove(ok(cb2), [ ], [ok(cb2)]) <- true 9 -> 8

dprove(connected_to(w6, w5), [ ], [ok(cb2)]) <- true 8 -> 6

?dprove(live(w5), [ok(cb2)], D) 4 10
dprove(G, D, [G| D]) <- delay(G)
dprove(live(w5), [ok(cb2)], [live(w5)| [ok(cb2)]]) <- delay(live(w5))

?dprove(live(w5), [ok(cb2)], D) 4 11
dprove(H, D11, D12) <- (H <= B) ∧dprove(B, D11, D12)
dprove(live(w5), [ok(cb2)], D) <- (live(w5) <= B) ∧dprove(B, [ok(cb2)], D)

# Proof steps using dprove (4)

? (live(w5) <= B) 11 12
live(W2) <= connected_to(W2, W3) & live(W3)
live(w5) <= connected_to(w5, W3) & live(W3)

?dprove((connected_to(w5, W3) & live(W3)), [ok(cb2)], ) 4 13
dprove((A & B), D13, D15) <- dprove(A, D13, D14) ∧dprove(B, D14, D15)
dprove((connected_to(w5, W3) & live(W3)), [ok(cb2)], D) <-
dprove(connected_to(w5, W3), [ok(cb2)], D14) ∧dprove(live(W3), D14, D)

?dprove(connected_to(w5, W3), [ok(cb2)], D14) 13 14
dprove(G, D16, [G| D16]) <- delay(G)
dprove(connected_to(w5, W3), [ok(cb2)], [connected_to(w5, W3)| [ok(cb2)]]) <-
delay(connected_to(w5, W3))

?dprove(connected_to(w5, W3), [ok(cb2)], D14) 13 15
dprove(H, D17, D18) <- (H <= B) ∧dprove(B, D17, D18)
dprove(connected_to(w5, W3), [ok(cb2)], D14) <- (connected_to(w5, W3) <= B)
∧dprove(B, [ok(cb2)], D14)

# Proof steps using dprove (5)

? (connected_to(w5, W3) <= B) 15 16
connected_to(w5, outside) <= ok(outside_connection)
connected_to(w5, outside) <= ok(outside_connection)

?dprove(ok(outside_connection), [ok(cb2)], D14) 15 17
dprove(G, D19, [G| D19]) <- delay(G)
dprove(ok(outside_connection), [ok(cb2)], [ok(outside_connection)| [ok(cb2)]])
<- delay(ok(outside_connection))

?delay(ok(outside_connection)) 17 18
delay(ok(X))
delay(ok(outside_connection))

dprove(ok(outside_connection), [ok(cb2)], [ok(outside_connection)| [ok(cb2)]])
<- true 18 -> 17

# Proof steps using dprove (6)

dprove(connected_to(w5, outside), [ok(cb2)], [ok(outside_connection), ok(cb2)]) <-
true 17 -> 15

?dprove(live(outside), [ok(outside_connection), ok(cb2)], D) 13 19
dprove(G, D20, [G| D20]) <- delay(G)
dprove(live(outside), [ok(outside_connection), ok(cb2)], [live(outside)| D20]) <-
delay(live(outside))

?dprove(live(outside), [ok(outside_connection), ok(cb2)], D) 13 20
dprove(H, D21, D22) <- (H <= B) ∧dprove(B, D21, D22)
dprove(live(outside), [ok(outside_connection), ok(cb2)], D2) <- (live(outside) <= B)
∧dprove(B, [ok(outside_connection), ok(cb2)], D)

? (live(outside) <= B) 20 21
live(outside) <= true
live(outside) <= true

# Proof steps using dprove (7)

?dprove(true, [ok(outside_connection), ok(cb2)], D) 20 22
dprove(true, D23, D23)
dprove(true, [ok(outside_connection), ok(cb2)], [ok(outside_connection), ok(cb2)])

dprove((connected_to(w5, outside) & live(outside)), [ok(cb2)],
[ok(outside_connection), ok(cb2)]) <- true 22 -> 20

dprove(live(outside), [ok(outside_connection), ok(cb2)], [ok(outside_connection),
ok(cb2)]) <- true 20 -> 13

dprove(live(w5), [ok(cb2)], [ok(outside_connection), ok(cb2)]) <- true 13 -> 11

dprove((connected_to(w6, w5) & live(w5)), [ ], [ok(outside_connection), ok(cb2)])
<- true 11 -> 4

dprove(live(w6), [ ], [ok(outside_connection), ok(cb2)]) <- true 4 -> 2

dprove(live(w6), [ ], [ok(outside_connection), ok(cb2)]) <- true 2 -> 0

# Meta-interpreter that builds a proof tree

% $hprove(G, T)$ is true if $G$ can be proved from the base-level

% KB, with proof tree $T$.

$$hprove(true, true).$$

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

$$\quad hprove(A, L) \wedge$$

$$\quad hprove(B, R).$$

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

$$\quad (H \Leftarrow B) \wedge$$

$$\quad hprove(B, T).$$