Variables

- Variables are **universally quantified** in the scope of a clause.
- A **variable assignment** is a function from variables into the domain.
- Given an interpretation and a variable assignment,
  each term denotes an individual and
  each clause is either true or false.
- A clause containing variables is true in an interpretation if it is true **for all** variable assignments.

Queries and Answers

A query is a way to ask if a body is a logical consequence of the knowledge base:

$$?b_1 \wedge \cdots \wedge b_m.$$

An answer is either

- an instance of the query that is a logical consequence of the knowledge base KB, or
- no if no instance is a logical consequence of KB.

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \land in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	*no*
$?in(kim, r023).$	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B)$.	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building)$.	*no*
$?in(kim, r023)$.	*no*
$?in(kim, B)$.	

Example Queries

$$KB = \begin{cases} in(kim, r123). \\ part\_of(r123, cs\_building). \\ in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z). \end{cases}$$

Query	Answer
$?part\_of(r123, B).$	$part\_of(r123, cs\_building)$
$?part\_of(r023, cs\_building).$	*no*
$?in(kim, r023).$	*no*
$?in(kim, B).$	$in(kim, r123)$ $in(kim, cs\_building)$

Logical Consequence

Atom g is a logical consequence of KB if and only if:

- g is a fact in KB, or
- there is a rule

$$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  in KB such that each b_i is a logical consequence of KB.

Debugging false conclusions

To debug answer g that is false in the intended interpretation:

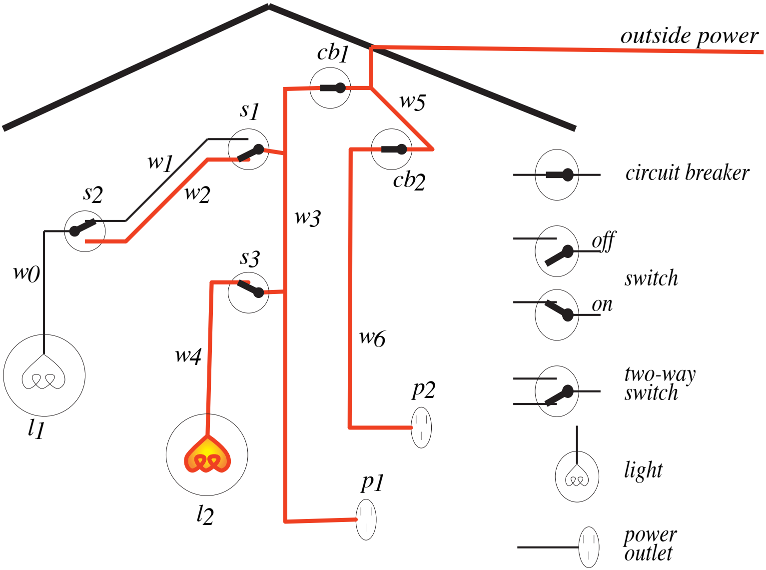
- If g is a fact in KB, this fact is wrong.
- Otherwise, suppose g was proved using the rule:

  $$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

  where each b_i is a logical consequence of KB.

  - If each b_i is true in the intended interpretation, this clause is false in the intended interpretation.
  - If some b_i is false in the intended interpretation, debug b_i.

Electrical Environment



Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ *yes*
$?light(l_6)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

$?light(l_1)$. $\Longrightarrow$ *yes*
$?light(l_6)$. $\Longrightarrow$ *no*
$?up(X)$. $\Longrightarrow$

Axiomatizing the Electrical Environment

% $light(L)$ is true if L is a light
$light(l_1)$. $light(l_2)$.
% $down(S)$ is true if switch S is down
$down(s_1)$. $up(s_2)$. $up(s_3)$.
% $ok(D)$ is true if D is not broken
$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

?$light(l_1)$. $\Longrightarrow$ *yes*
?$light(l_6)$. $\Longrightarrow$ *no*
?$up(X)$. $\Longrightarrow$ $up(s_2)$, $up(s_3)$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \quad \Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \quad \Longrightarrow \quad W = w_1$

$?connected\_to(w_1, W). \quad \Longrightarrow$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \implies W = w_1$

$?connected\_to(w_1, W). \implies no$

$?connected\_to(Y, w_3). \implies$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \implies W = w_1$

$?connected\_to(w_1, W). \implies no$

$?connected\_to(Y, w_3). \implies Y = w_2,\ Y = w_4,\ Y = p_1$

$?connected\_to(X, W). \implies$

$connected\_to(X, Y)$ is true if component X is connected to Y

$$connected\_to(w_0, w_1) \leftarrow up(s_2).$$
$$connected\_to(w_0, w_2) \leftarrow down(s_2).$$
$$connected\_to(w_1, w_3) \leftarrow up(s_1).$$
$$connected\_to(w_2, w_3) \leftarrow down(s_1).$$
$$connected\_to(w_4, w_3) \leftarrow up(s_3).$$
$$connected\_to(p_1, w_3).$$

$?connected\_to(w_0, W). \quad \Longrightarrow \quad W = w_1$

$?connected\_to(w_1, W). \quad \Longrightarrow \quad no$

$?connected\_to(Y, w_3). \quad \Longrightarrow \quad Y = w_2,\ Y = w_4,\ Y = p_1$

$?connected\_to(X, W). \quad \Longrightarrow \quad X = w_0, W = w_1, \ldots$

% $lit(L)$ is true if the light L is lit

$$lit(L) \leftarrow light(L) \wedge ok(L) \wedge live(L).$$

% $live(C)$ is true if there is power coming into C

$$live(Y) \leftarrow$$
$$connected\_to(Y, Z) \wedge$$
$$live(Z).$$
$$live(outside).$$

This is a recursive definition of *live*.

Recursion and Mathematical Induction

$$above(X, Y) \leftarrow on(X, Y).$$
$$above(X, Y) \leftarrow on(X, Z) \wedge above(Z, Y).$$

This can be seen as:

- Recursive definition of *above*: prove *above* in terms of a base case (*on*) or a simpler instance of itself; or
- Way to prove *above* by mathematical induction: the base case is when there are no blocks between X and Y, and if you can prove *above* when there are n blocks between them, you can prove it when there are $n + 1$ blocks.

Limitations

Suppose you had a database using the relation:

$$enrolled(S, C)$$

which is true when student S is enrolled in course C.
You can't define the relation:

$$empty\_course(C)$$

which is true when course C has no students enrolled in it.
This is because $empty\_course(C)$ doesn't logically follow from a set of *enrolled* relations. There are always models where someone is enrolled in a course!