# Chapters 2&3: A Representation and Reasoning System

- **Lecture 1** Representation and Reasoning Systems. Datalog.
- **Lecture 2** Semantics.
- **Lecture 3** Variables, queries and answers, limitations.
- **Lecture 4** Proofs. Soundness and completeness.
- **Lecture 5** SLD resolution.
- **Lecture 6** Proofs with variables. Function Symbols.

©David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1997

# 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.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Remarks on “Semantics of Variables”

- Datalog
  - Assignment $\rho: V \rightarrow D$
  - $\rho$ assigns to each variable one element of the domain.
- Schöning / F2-Vorlesung
  - The interpretation (mapping) $I$ treats the variables, too, i.e. beyond constants, predicate and function symbols.
- LOS (Logics and Semantics) course
  - Assignment $A: V \rightarrow D$

© Christopher. Habel – University of Hamburg – 2001

# Evaluation of quantified expressions

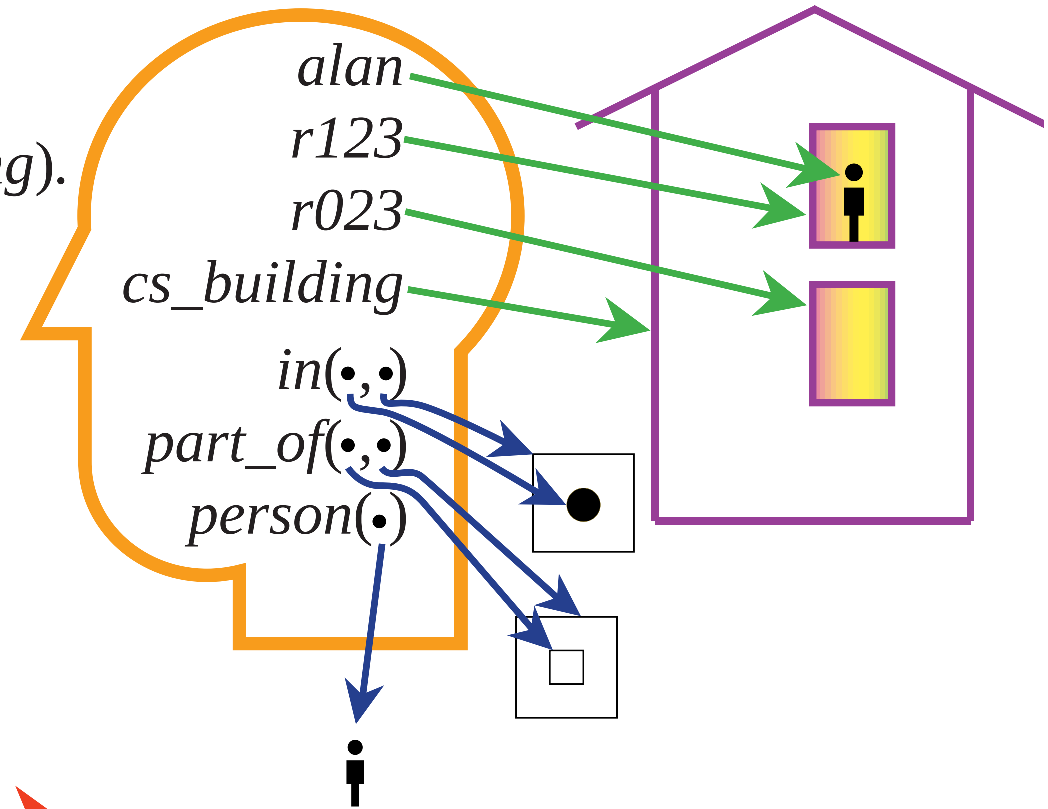
Example domain (cg. Figure 2.1) :

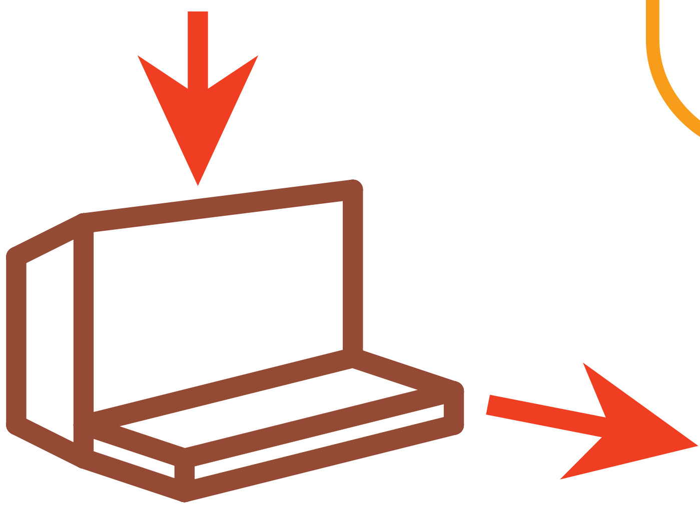
- *part_of (X, Y)* $\leftarrow$ *in (X, Y)*
  is *FALSE* in the interpretation (cf. Lect. 2.1&2)
  Assignment: $\rho$: $X \rightarrow$ alan´ $Y \rightarrow$ r123´

- *in (X, Y)* $\leftarrow$ *part_of (Z, Y)* $\wedge$ *in (X, Z)*
  is *TRUE* in the interpretation (cf. Lect. 2.1&2),
  since all assignments make the clause true.

# Role of Semantics in an RRS

*in*(*alan*,*r123*).
*part_of*(*r123*,*cs_building*).
*in*(*X*,*Y*) ← 
  *part_of*(*Z*,*Y*) ∧
  *in*(*X*,*Z*).



*in*(*alan*,*cs_building*)

# 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*.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Example Queries

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

| Query | Answer |
|---|---|
| $?part\_of(r123, B).$ | $part\_of(r123, cs\_building)$ |
| $?part\_of(r023, cs\_building).$ | $no$ |
| $?in(alan, r023).$ | $no$ |
| $?in(alan, B).$ | $in(alan, r123)$ |
| | $in(alan, cs\_building)$ |

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Variables in Questions and Answers

- Example. 2.9 (Robot's world):
  *two_doors_east (E, W)* $\leftarrow$
  *imm_east (E, M)* $\wedge$ *imm_east (M, W)*
- Example. 2.12:
  query: ?*two_doors_east (R, r107)*
  - The relevant instances:
    *imm_east (r111, r109)* $\wedge$ *imm_east (r109, r107)*
- (specific) Answer clause
  *yes (R)* $\leftarrow$ *two_doors_east (R, r107)*
- (general) Answer clause
  $yes\ (V_1, \ldots, V_n) \leftarrow Body$

© Christopher. Habel – University of Hamburg – 2001

# 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 $KB$.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# 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:

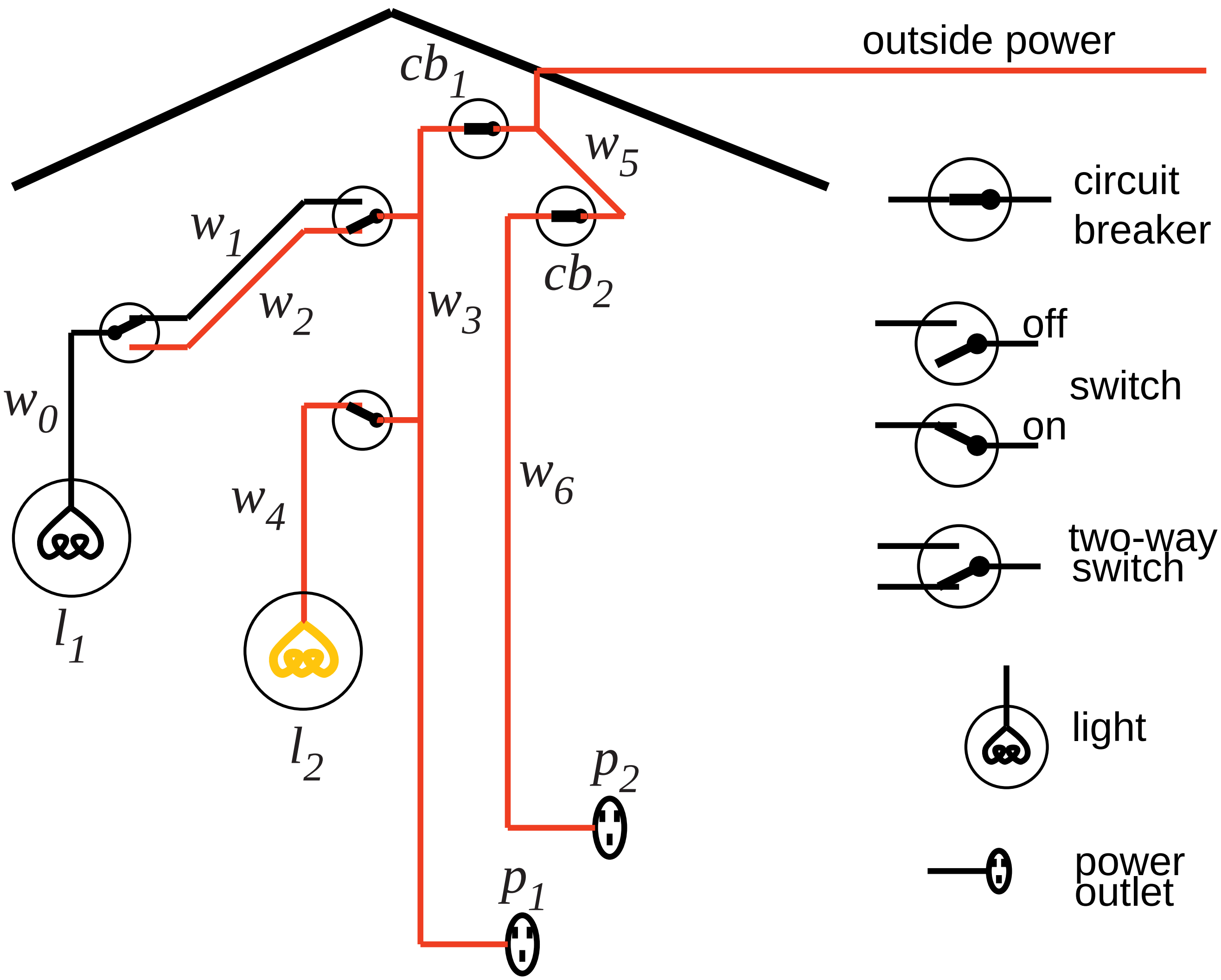
$$g \leftarrow b_1 \wedge \ldots \wedge b_k$$

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 $b_i$.


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Domain for Diagnostic Assistant




© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# House Wiring: Ontology and intended interpretations

- Types of things:
  *Lights, Wires, Switches, Circuit breakers, Power outlets*
- | | |
  |---|---|
  | *light (L)* | L is a light |
  | *lit (L)* | the light L is lit, and emitting light |
  | *live (W)* | W is live (power coming into W) |
  | *up (S), down (S)* | switch S is up / down |
  | *ok (E)* | E is not blown (lights, circ. br.) |
  | *connected_to (X,Y)* | components X & Y are connected |

# 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

$ok(l_1)$. $ok(l_2)$. $ok(cb_1)$. $ok(cb_2)$.

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

$?light(l_6)$. $\Longrightarrow$ $no$

$?up(X)$. $\Longrightarrow$ $up(s_2), up(s_3)$

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

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

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

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

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

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

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


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

% $lit(L)$ 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*.

# Exercise B : To discussed in Lecture 2. 5&6

- Extend the House Wiring domain:
  i. Introduce an additional light of the class “desk lamp”, which is connected via a power outlet.
  Which facts and rules have to be added to the knowledge base?
  ii. Change the representation of switches by introducing the status of connecting input and output wire - instead of using *up* and *down*.
  Which types of new individuals have to be introduced into the domain?

© Christopher. Habel – University of Hamburg – 2001

# 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.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# 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!


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Proofs

- A proof is a mechanically derivable demonstration that a formula logically follows from a knowledge base.
- Given a proof procedure, $KB \vdash g$ means $g$ can be derived from knowledge base $KB$.
- Recall $KB \models g$ means $g$ is true in all models of $KB$.
- A proof procedure is sound if $KB \vdash g$ implies $KB \models g$.
- A proof procedure is complete if $KB \models g$ implies $KB \vdash g$.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Bottom-up Ground Proof Procedure

One **rule of derivation,** a generalized form of *modus ponens*:

> If "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" is a clause in the knowledge base, and each $b_i$ has been derived, then $h$ can be derived.

You are **forward chaining** on this clause.

(This rule also covers the case when $m = 0$.)

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Bottom-up proof procedure

$KB \vdash g$ if $g \in C$ at the end of this procedure:

$C := \{\}$;
**repeat**
    **select** clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" in $KB$ such that
        $b_i \in C$ for all $i$, and
        $h \notin C$;
    $C := C \cup \{h\}$
**until** no more clauses can be selected.

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Example 2.14: Bottom up proof procedure

Knowledge Base:

$a \leftarrow b \wedge c$

$b \leftarrow d \wedge e$

$b \leftarrow g \wedge e$

$c \leftarrow e$

$d$

$e$

$f \leftarrow a \wedge g$

Consequence set *C*

| | |
|---|---|
| 1 | $\{\}$ |
| 2 | $\{d\}$ |
| 3 | $\{d, e\}$ |
| 4 | $\{b, d, e\}$ |
| 5 | $\{b, c, d, e\}$ |
| 6 | $\{a, b, c, d, e\}$ |

© Christopher. Habel – University of Hamburg – 2001

# Nondeterministic Choice

- Don't-care nondeterminism If one selection doesn't lead to a solution, there is no point trying other alternatives. select
- Don't-know nondeterminism If one choice doesn't lead to a solution, other choices may. choose


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Example

$a \leftarrow b \wedge c.$

$a \leftarrow e \wedge f.$

$b \leftarrow f \wedge k.$

$c \leftarrow e.$

$d \leftarrow k.$

$e.$

$f \leftarrow j \wedge e.$

$f \leftarrow c.$

$j \leftarrow c.$


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Soundness of bottom-up proof procedure

If $KB \vdash g$ then $KB \models g$.

Suppose there is a $g$ such that $KB \vdash g$ and $KB \not\models g$.

Let $h$ be the first atom added to $C$ that's not true in every model of $KB$. Suppose $h$ isn't true in model $I$ of $KB$.
There must be a clause in $KB$ of form

$$h \leftarrow b_1 \wedge \ldots \wedge b_m$$

Each $b_i$ is true in $I$. $h$ is false in $I$. So this clause is false in $I$.
Therefore $I$ isn't a model of $KB$.

Contradiction: thus no such $g$ exists.


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Fixed Point

The $C$ generated at the end of the bottom-up algorithm is called a fixed point.

Let $I$ be the interpretation in which every element of the fixed point is true and every other atom is false.

$I$ is a model of $KB$.

Proof: suppose $h \leftarrow b_1 \wedge \ldots \wedge b_m$ in $KB$ is false in $I$. Then $h$ is false and each $b_i$ is true in $I$. Thus $h$ can be added to $C$. Contradiction to $C$ being the fixed point.

$I$ is called a Minimal Model.


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998

# Completeness

**If $KB \models g$ then $KB \vdash g$.**

Suppose $KB \models g$. Then $g$ is true in all models of $KB$.

Thus $g$ is true in the minimal model.

Thus $g$ is generated by the bottom up algorithm.

Thus $KB \vdash g$.


© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1998