Using Definite Knowledge

Representation and Reasoning System

- RRS
  - Syntax
  - Semantics
  - Proof procedures (Bottom-up,Top-down)
- Implementation of RRS
  - Parser
  - Reasoning procedure (implementation of the proof procedure)
- How to use the RRS?
  - Representational methodology (user manual, created by design and through experience and)
  - Programming methodology (specification of the logic of the domain and specification of the control for finding a solution)

Different Views of Datalog

- A database language (deductive database)
- A question-answering system
- A programming language
- A representation and reasoning system

How to use the RRS (Datalog) to represent a domain, answer questions, and solve problems in the domain?

Applications of RRS

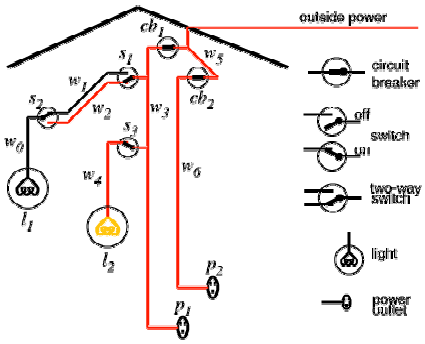
Assumptions

- *Intended interpretation* for symbols of the domain
- *Clauses* can be viewed as *statements* about the domain

Important design decision

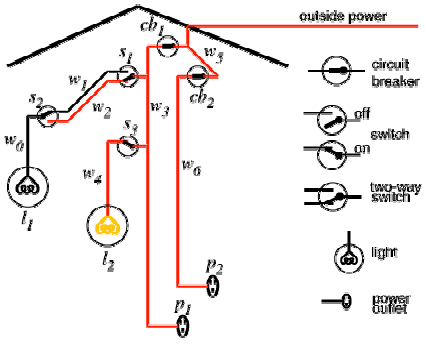
- What are the concepts and individuals?
- What level of detail?
- Is each clause true in the intended interpretation?
- Do the rules for the predicates cover all the cases?

House Wiring



- Goal: determine whether the light is on/off? (based on positions & status of other components)
- Not interested in color of the wire, the type of the switches, the length of the wire, etc.

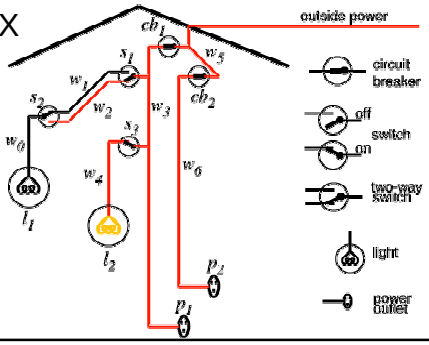
House Wiring



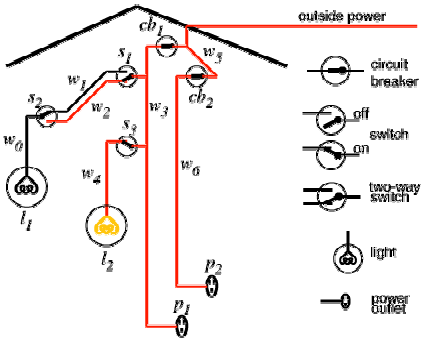
- What level of abstraction?
  - Commonsense-level
- What are the individuals?
  - Wire
  - Switch
  - Light
  - Power outlet

House Wiring

- Choose predicates:
  - light(L) is true if the individual denoted by L is a light
  - lit(L) is true if the light L is lit (on)
  - live(W) is true if the there is power coming into W
  - up(S)/down(S) is true if switch S is up/down
  - ok(E) if true E is not defected (circuit breaker, light)
  - connected_to(X,Y) is true if X is connected to Y
- What is the possible atoms?



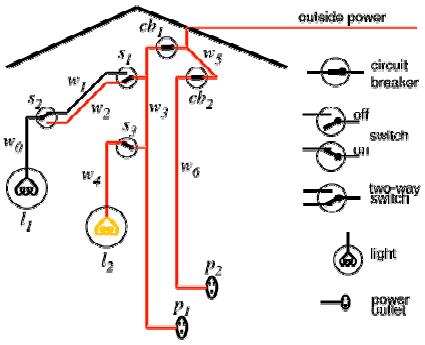
House Wiring



Facts

light(l1)
light(l2)
down(s1)
up(s2)
up(s3)
ok(l1)
ok(l2)
ok(cb1)
ok(cb2)

House Wiring



Facts

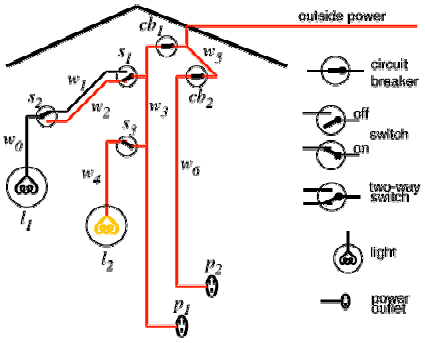
light(l1)
light(l2)
down(s1)
up(s2)
up(s3)
ok(l1)
ok(l2)
ok(cb1)
ok(cb2)

Questions

? light(l1)
? up(X)
? ok(X)

Answers ?

House Wiring

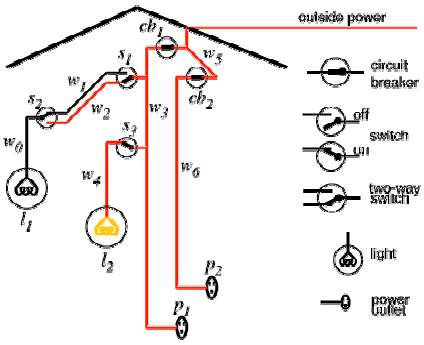


? connected_to(l1,W)
? connected_to(X,w3)

Facts/Clauses

connected_to(l1,w0).
connected_to(w0,w1)←up(s2).
connected_to(w0,w2)←down(s2).
connected_to(w1,w3)←up(s1).
connected_to(w2,w3)←down(s1).
connected_to(l2,w3).
connected_to(w4,w3)←up(s3).
connected_to(p1,w3).
connected_to(w3,w5)←ok(cb1).
connected_to(p2,w6).
connected_to(w6,w5)←ok(cb2).
connected_to(w5,outside).

House Wiring



Clauses

lit(L) ← light(L) ∧ ok(L) ∧ live(L).

live(X) ← connected_to(X,Y)∧ live(Y).

live(outside).

? live(W)
? lit(X)

W=w2, W=w3, W=w5, W=w6, W=p1, W=p2, W=l2, W=w4, W=p2
W=outside
X=l2

Database Operations

- Datalog can be used as a database language
- Each tuple in a relational database could be viewed as a ground fact in a knowledger base

Database

Relational database

312	comp_science
322	comp_science
315	math
371	physics

Ground facts

department(312,comp_science)
department(322,comp_science)
department(315,math)
department(371,physics)

```
% course(C) is true if C is a university course
course(312).
course(322).
course(315).
course(371).

% department(C,D) is true if course C is offered in
      department D.
department(312,comp_science).
department(322,comp_science).
department(315,math).
department(371,physics).

% student(S) is true if S is a student
student(mary).
student(jane).
student(john).
student(harold).

% female(P) is true if person P is female
female(mary).
female(jane).

% enrolled(S,C) is true if student S is enrolled in
      course C
enrolled(mary,322).
enrolled(mary,312).
enrolled(john,322).
enrolled(john,315).
enrolled(harold,322).
enrolled(mary,315).
enrolled(jane,312).
enrolled(jane,322).

% cs_course(C) is true if course C is offered in CS

cs_course(C) :- department(C,comp_science).

% math_course(C) is true if course C is offered in ..

math_course(C) :- department(C,math).

% cs_or_math_course(C) is true if course C is
      offered in CS or math

cs_or_math_course(C) :- cs_course(C).
cs_or_math_course(C) :- math_course(C).

% in_dept(S,D) is true if student S is enrolled
% in a course offered in deparment D

in_dept(S,D) :- enrolled(S,C), department(C,D).


% example query
% ? enrolled(S,C), department(C,D).
```

Database Operations

- Selection (some tuples of a database)
- Union (tuples from different databases)
- Join (attributes from different databases)
- Projection (some attributes of a database)

Selection

- Use constants in clauses or defining a condition
- Example

  cs_course(X)←department(X,comp_science).
  math_course(X)←department(X,math).

  ?cs_course(X)
  ?math_course(X)

Union

- Use multiple rules with the same head
- Example

  cs_course(X)←department(X,comp_science).
  math_course(X)←department(X,math).

  cs_or_math_course(X) ← math_course(X).
  cs_or_math_course(X) ← cs_course(X).

  ? cs_or_math_course(X)

Join

- Use the same variable for the join attribute in a conjunction of two relations
- Example

  ?enrolled(S,C) ∧ department(C,D)

  (join of the *enrolled* and *department* relations on the course attribute)

Projection

- Use variables that do not occur in the head
- Example

  course(X) ← department(X,D).

  (projection onto the course)

  in_dept(S,D) ← enrolled(S,C) ∧ department(C,D).

  (projection onto the student and department attributes of the join of the *enrolled* and *department* relations)

Recursion

- Recursion: define a predicate in terms of *simpler* instances of itself
- Example

  west(R1,R2) ← imm_west(R1,R2).

  west(R1,R2) ← imm_west(R1,R) ∧ west(R,R2).

  imm_west(r101,r103).

  ...

Recursion

- Idea: *well-founded ordering* among the instances of the relations such that
  - higher level relation is defined in terms of elements in lower levels
  - the lowest level relation is defined by clauses without body.
- Example

  west(R1,R2) ← imm_west(R1,R2).

  west(R1,R2) ← imm_west(R1,R) ∧ west(R,R2).

  imm_west(r101,r103).

  ...

Recursion and Mathematical Induction

Recursion	Mathematical Induction
• Top-down (from higher level to lower)	• Bottom-up (e.g. from n to n+1)
• Simplest level: fact	• Base case: n=0