

# Simple language: propositional definite clauses

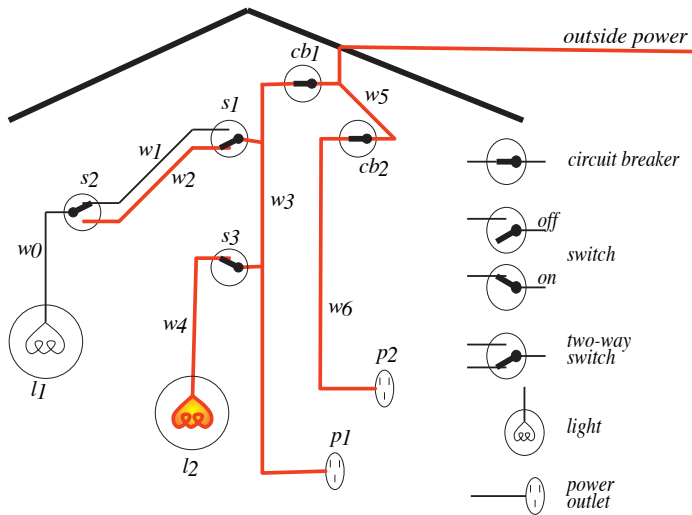
Propositional definite clauses are a restricted form of propositions that can't represent disjunction of atoms:

- A **body** is either
  - ▶ an atom or
  - ▶ the form  $b_1 \wedge b_2$  where  $b_1$  and  $b_2$  are bodies.
- A **definite clause** is either
  - ▶ an **atomic fact**: an atom or
  - ▶ a **rule**: of the form  $h \leftarrow b$  where  $h$  is an atom and  $b$  is a body.

An atomic fact is treated as a rule with an empty body.

- A **knowledge base** or **logic program** is a set of definite clauses.
- A **query** is a body that is asked of a knowledge base.

# Electrical Environment



# Representing the Electrical Environment

*light\_l1.*

*light\_l2.*

*down\_s1.*

*up\_s2.*

*up\_s3.*

*ok\_l1.*

*ok\_l2.*

*ok\_cb1.*

*ok\_cb2.*

*live\_outside.*

*lit\_l1*  $\leftarrow$  *live\_w0*  $\wedge$  *ok\_l1*

*live\_w0*  $\leftarrow$  *live\_w1*  $\wedge$  *up\_s2.*

*live\_w0*  $\leftarrow$  *live\_w2*  $\wedge$  *down\_s2.*

*live\_w1*  $\leftarrow$  *live\_w3*  $\wedge$  *up\_s1.*

*live\_w2*  $\leftarrow$  *live\_w3*  $\wedge$  *down\_s1.*

*lit\_l2*  $\leftarrow$  *live\_w4*  $\wedge$  *ok\_l2.*

*live\_w4*  $\leftarrow$  *live\_w3*  $\wedge$  *up\_s3.*

*live\_p1*  $\leftarrow$  *live\_w3.*

*live\_w3*  $\leftarrow$  *live\_w5*  $\wedge$  *ok\_cb1.*

*live\_p2*  $\leftarrow$  *live\_w6.*

*live\_w6*  $\leftarrow$  *live\_w5*  $\wedge$  *ok\_cb2.*

*live\_w5*  $\leftarrow$  *live\_outside.*

- 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  $KB \models g$ .
  - ▶ If a sound proof procedure produces a result, the result is correct.
- A proof procedure is **complete** if  $KB \models g$  implies  $KB \vdash g$ .
  - ▶ A complete proof procedure can produce all results.

# Bottom-up Proof Procedure

One **rule of derivation**, a generalized form of *modus ponens*:  
*If " $h \leftarrow b_1 \wedge \dots \wedge b_m$ " is a clause in the knowledge base,*  
*and each  $b_i$  has been derived, then  $h$  can be derived.*

This is **forward chaining** on this clause.

(An atomic fact is treated as a clause with empty body ( $m = 0$ ).)

# Bottom-up proof procedure

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

$C := \{\}$ ;

**repeat**

**select** clause " $h \leftarrow b_1 \wedge \dots \wedge b_m$ " 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.

# Example

$$a \leftarrow b \wedge c.$$

$$a \leftarrow e \wedge f.$$

$$b \leftarrow f \wedge k.$$

$$c \leftarrow e.$$

$$d \leftarrow k.$$

$$e.$$

$$f \leftarrow j \wedge e.$$

$$f \leftarrow c.$$

$$j \leftarrow c.$$

# 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  $KB \not\models g$ .
- Then there must be a first atom added to  $C$  that isn't true in every model of  $KB$ . Call it  $h$ .  
Suppose  $h$  isn't true in model  $I$  of  $KB$ .
- $h$  was added to  $C$ , so there must be a clause in  $KB$

$$h \leftarrow b_1 \wedge \dots \wedge b_m$$

where each  $b_i$  is in  $C$ , and so true in  $I$ .

$h$  is false in  $I$  (by assumption)

So this clause is false in  $I$ .

Therefore  $I$  isn't a model of  $KB$ .

- Contradiction. Therefore there cannot be such a  $g$ .



- 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.
- Claim:  $I$  is a model of  $KB$ .  
Proof: suppose  $h \leftarrow b_1 \wedge \dots \wedge b_m$  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**.

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 in the fixed point.
- Thus  $g$  is generated by the bottom up algorithm.
- Thus  $KB \vdash g$ .

# Top-down Definite Clause Proof Procedure

Idea: search backward from a query to determine if it is a logical consequence of  $KB$ .

An **answer clause** is of the form:

$$yes \leftarrow a_1 \wedge a_2 \wedge \dots \wedge a_m$$

The **SLD Resolution** of this answer clause on atom  $a_i$  with the clause:

$$a_i \leftarrow b_1 \wedge \dots \wedge b_p$$

is the answer clause

$$yes \leftarrow a_1 \wedge \dots \wedge a_{i-1} \wedge b_1 \wedge \dots \wedge b_p \wedge a_{i+1} \wedge \dots \wedge a_m.$$

An atomic fact in the knowledge base is considered as a clause where  $p = 0$ .

- An **answer** is an answer clause with  $m = 0$ .  
That is, it is the answer clause  $\text{yes} \leftarrow$ .
- A **derivation** of query “ $?q_1 \wedge \dots \wedge q_k$ ” from  $KB$  is a sequence of answer clauses  $\gamma_0, \gamma_1, \dots, \gamma_n$  such that
  - ▶  $\gamma_0$  is the answer clause  $\text{yes} \leftarrow q_1 \wedge \dots \wedge q_k$
  - ▶  $\gamma_i$  is obtained by resolving  $\gamma_{i-1}$  with a clause in  $KB$
  - ▶  $\gamma_n$  is an answer.

To solve the query  $?q_1 \wedge \dots \wedge q_k$ :

$ac := \text{“yes} \leftarrow q_1 \wedge \dots \wedge q_k\text{”}$

**repeat**

**select** atom  $a_i$  from the body of  $ac$

**choose** clause  $C$  from  $KB$  with  $a_i$  as head

    replace  $a_i$  in the body of  $ac$  by the body of  $C$

**until**  $ac$  is an answer.

# 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

## Example: successful derivation

$a \leftarrow b \wedge c.$

$c \leftarrow e.$

$f \leftarrow j \wedge e.$

$a \leftarrow e \wedge f.$

$d \leftarrow k.$

$f \leftarrow c.$

$b \leftarrow f \wedge k.$

$e.$

$j \leftarrow c.$

Query: ?a

$\gamma_0 : \text{yes} \leftarrow a$

$\gamma_1 : \text{yes} \leftarrow e \wedge f$

$\gamma_2 : \text{yes} \leftarrow f$

$\gamma_3 : \text{yes} \leftarrow c$

$\gamma_4 : \text{yes} \leftarrow e$

$\gamma_5 : \text{yes} \leftarrow$

## Example: failing derivation

$a \leftarrow b \wedge c.$

$c \leftarrow e.$

$f \leftarrow j \wedge e.$

$a \leftarrow e \wedge f.$

$d \leftarrow k.$

$f \leftarrow c.$

$b \leftarrow f \wedge k.$

$e.$

$j \leftarrow c.$

Query: ?a

$\gamma_0 : \text{yes} \leftarrow a$

$\gamma_1 : \text{yes} \leftarrow b \wedge c$

$\gamma_2 : \text{yes} \leftarrow f \wedge k \wedge c$

$\gamma_3 : \text{yes} \leftarrow c \wedge k \wedge c$

$\gamma_4 : \text{yes} \leftarrow e \wedge k \wedge c$

$\gamma_5 : \text{yes} \leftarrow k \wedge c$



# Search Graph for SLD Resolution

$a \leftarrow b \wedge c.$	$a \leftarrow g.$
$a \leftarrow h.$	$b \leftarrow j.$
$b \leftarrow k.$	$d \leftarrow m.$
$d \leftarrow p.$	$f \leftarrow m.$
$sf \leftarrow p.$	$g \leftarrow m.$
$g \leftarrow f.$	$k \leftarrow m.$
$h \leftarrow m.$	$p.$
$?a \wedge d$	

