Propositions and inference

Chapter 5

David Poole and Alan Mackworth

Propositions

- A proposition is a sentence, written in a language, that has a truth value – it is true or false – in a world. A proposition is built from atomic propositions (atoms) and logical connectives.
- Propositions can be built from simpler propositions using logical connectives.

Propositional Calculus Syntax

- An atomic proposition – atom – is a symbol, written as sequences of letters, digits, and the underscore (_) and start with a lower-case letter.
  E.g., a, ai_is_fun, lit_l_1, $live\_outside$, $mimsy$, $sunny$.
- A proposition or logical formula is either
  - an atomic proposition or
  - a compound proposition of the form

$\neg p$	"not p"	negation of p
$p \wedge q$	"p and q"	conjunction of p and q
$p \vee q$	"p or q"	disjunction of p and q
$p \rightarrow q$	"p implies q"	implication of q from p
$p \leftarrow q$	"p if q"	implication of p from q
$p \leftrightarrow q$	"p if and only if q"	equivalence of p and q
$p \oplus q$	"p XOR q"	exclusive-or of p and q

    where p and q are propositions.
- The operators $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftarrow$, $\leftrightarrow$, and $\oplus$ are logical connectives.

Semantics of the Propositional Calculus

- An interpretation – or possible world – is an assignment of true or false to each variable.
- An interpretation is defined by function π that maps atoms to $\{true, false\}$.
  If $\pi(a) = true$, atom a is true in the interpretation.
  If $\pi(a) = false$, atom a is false in the interpretation.
- Truth of a compound proposition in an interpretation is defined in terms of the truth of its components:

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftarrow q$	$p \leftrightarrow q$	$p \oplus q$
true	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*true*	*false*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*	*false*	*true*
false	*false*	*true*	*false*	*false*	*true*	*true*	*true*	*false*

- Propositions can have differenet truth values in different interpretations.

Models and Logical Consequence

- A model of a set of clauses is an interpretation in which all the clauses are *true*.
- If KB is a set of propositions, proposition g is a logical consequence of KB, written $KB \models g$, if g is *true* in every model of KB.
- That is, $KB \models g$ if there is no interpretation in which KB is *true* and g is *false*.

Simple Example

$$KB = \begin{cases} apple\_eaten \leftarrow bird\_eats\_apple. \\ light\_on \leftarrow night. \\ night. \end{cases}$$

	$apple\_eaten$	$bird\_eats\_apple$	$light\_on$	$night$	model of KB?
I_1	$true$	$true$	$true$	$true$	
I_2	$false$	$false$	$false$	$false$	
I_3	$true$	$true$	$false$	$false$	
I_4	$false$	$false$	$true$	$true$	
I_5	$true$	$false$	$true$	$true$	

Simple Example

$$KB = \begin{cases} apple\_eaten \leftarrow bird\_eats\_apple. \\ light\_on \leftarrow night. \\ night. \end{cases}$$

	$apple\_eaten$	$bird\_eats\_apple$	$light\_on$	$night$	model of KB?
I_1	$true$	$true$	$true$	$true$	yes
I_2	$false$	$false$	$false$	$false$	no
I_3	$true$	$true$	$false$	$false$	no
I_4	$false$	$false$	$true$	$true$	yes
I_5	$true$	$false$	$true$	$true$	yes

Simple Example

$$KB = \begin{cases} apple\_eaten \leftarrow bird\_eats\_apple. \\ light\_on \leftarrow night. \\ night. \end{cases}$$

	$apple\_eaten$	$bird\_eats\_apple$	$light\_on$	$night$	model of KB?
I_1	$true$	$true$	$true$	$true$	yes
I_2	$false$	$false$	$false$	$false$	no
I_3	$true$	$true$	$false$	$false$	no
I_4	$false$	$false$	$true$	$true$	yes
I_5	$true$	$false$	$true$	$true$	yes

Which of $apple\_eaten$, $bird\_eats\_apple$, $light\_on$, $night$ logically follow from KB?

Simple Example

$$KB = \begin{cases} apple\_eaten \leftarrow bird\_eats\_apple. \\ light\_on \leftarrow night. \\ night. \end{cases}$$

	apple_eaten	*bird_eats_apple*	*light_on*	*night*	model of KB?
I_1	*true*	*true*	*true*	*true*	yes
I_2	*false*	*false*	*false*	*false*	no
I_3	*true*	*true*	*false*	*false*	no
I_4	*false*	*false*	*true*	*true*	yes
I_5	*true*	*false*	*true*	*true*	yes

Which of *apple_eaten*, *bird_eats_apple*, *light_on*, *night* logically follow from KB?

$KB \models light\_on$, $KB \models night$,

$KB \not\models apple\_eaten$, $KB \not\models bird\_eats\_apple$

Human's view of semantics

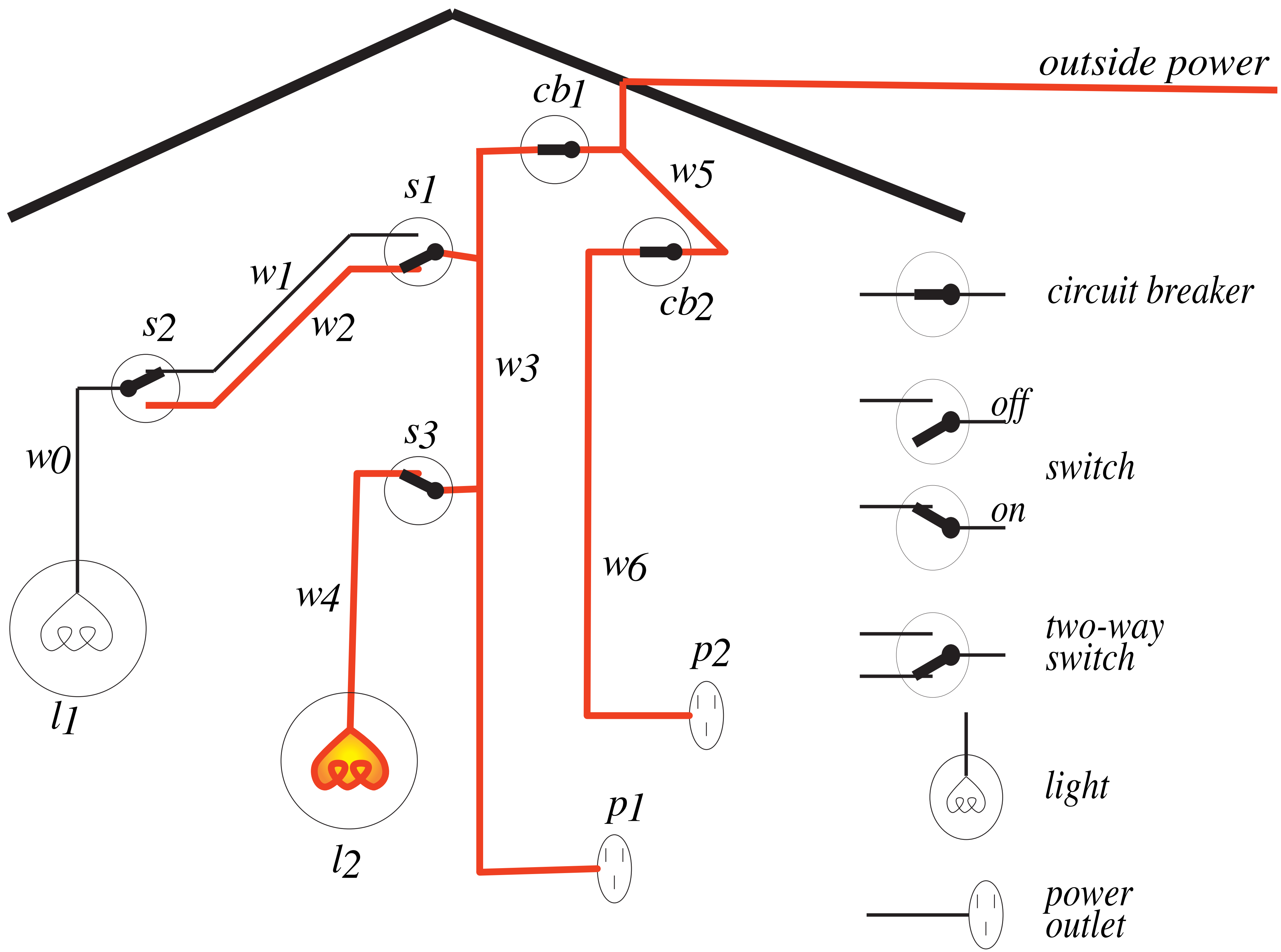
- Step 1 Begin with a task domain.
- Step 2 Choose atoms in the computer to denote propositions. These atoms have meaning to the KB designer.
- Step 3 Tell the system knowledge about the domain.
- Step 4 Ask the system questions.

— The system can tell you whether the question is a logical consequence.
— You can interpret the answer with the meaning associated with the atoms.

Computer's view of semantics

- The computer doesn't have access to the intended interpretation.
- All it knows is the knowledge base.
- The computer can determine if a formula is a logical consequence of KB.
- If $KB \models g$ then g must be true in the intended interpretation.
- If $KB \not\models g$ then there is a model of KB in which g is false. This could be the intended interpretation.

Electrical Environment



Role of semantics

In computer:

$$light2\_broken \leftarrow power\_in\_w\_3 \wedge sw\_3\_up \wedge unlit\_light2.$$

$$sw\_3\_up.$$

$$power\_in\_w\_3 \leftarrow power\_in\_p\_1.$$

$$unlit\_light2.$$

$$power\_in\_p\_1.$$

In user's mind:

- $light2\_broken$: light #2 is broken
- sw_3_up: switch 3 is up
- $power\_in\_w\_3$: there is power in wire 3
- $unlit\_light2$: light #2 isn't lit
- $power\_in\_p\_1$: outlet p_1 has power

Conclusion: $light2\_broken$

- The computer doesn't know the meaning of the symbols
- The user can interpret symbols using their meaning

Simple language: propositional definite clauses

Propositional definite clauses are a resticited form of propostions 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 qeury is a body that is asked of a knowledge base.

Representing the Electrical Environment

$light\_l_1.$

$light\_l_2.$

$down\_s_1.$

$up\_s_2.$

$up\_s_3.$

$ok\_l_1.$

$ok\_l_2.$

$ok\_cb_1.$

$ok\_cb_2.$

$live\_outside.$

$lit\_l_1 \leftarrow live\_w_0 \wedge ok\_l_1$

$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2.$

$live\_w_0 \leftarrow live\_w_2 \wedge down\_s_2.$

$live\_w_1 \leftarrow live\_w_3 \wedge up\_s_1.$

$live\_w_2 \leftarrow live\_w_3 \wedge down\_s_1.$

$lit\_l_2 \leftarrow live\_w_4 \wedge ok\_l_2.$

$live\_w_4 \leftarrow live\_w_3 \wedge up\_s_3.$

$live\_p_1 \leftarrow live\_w_3.$

$live\_w_3 \leftarrow live\_w_5 \wedge ok\_cb_1.$

$live\_p_2 \leftarrow live\_w_6.$

$live\_w_6 \leftarrow live\_w_5 \wedge ok\_cb_2.$

$live\_w_5 \leftarrow live\_outside.$

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 $KB \models g$.
  - 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 "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" 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 "$h \leftarrow b_1 \land \ldots \land b_m$" 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

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

  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.

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.
- Claim: 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.

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 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:

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

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

$$a_i \leftarrow b_1 \wedge \ldots \wedge b_p$$

is the answer clause

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

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

Derivations

- An answer is an answer clause with $m = 0$.
  That is, it is the answer clause $yes \leftarrow$.
- A derivation of query "$?q_1 \wedge \ldots \wedge q_k$" from KB is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  - γ_0 is the answer clause $yes \leftarrow q_1 \wedge \ldots \wedge q_k$
  - γ_i is obtained by resolving γ_{i-1} with a clause in KB
  - γ_n is an answer.

Top-down definite clause interpreter

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

> $ac :=$ "$yes \leftarrow q_1 \wedge \ldots \wedge q_k$"
> **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

$$a \leftarrow b \wedge c. \qquad a \leftarrow e \wedge f. \qquad b \leftarrow f \wedge k.$$

$$c \leftarrow e. \qquad d \leftarrow k. \qquad e.$$

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

Query: $?a$

$\gamma_0 : \quad yes \leftarrow a$

$\gamma_1 : \quad yes \leftarrow e \wedge f$

$\gamma_2 : \quad yes \leftarrow f$

$\gamma_3 : \quad yes \leftarrow c$

$\gamma_4 : \quad yes \leftarrow e$

$\gamma_5 : \quad yes \leftarrow$

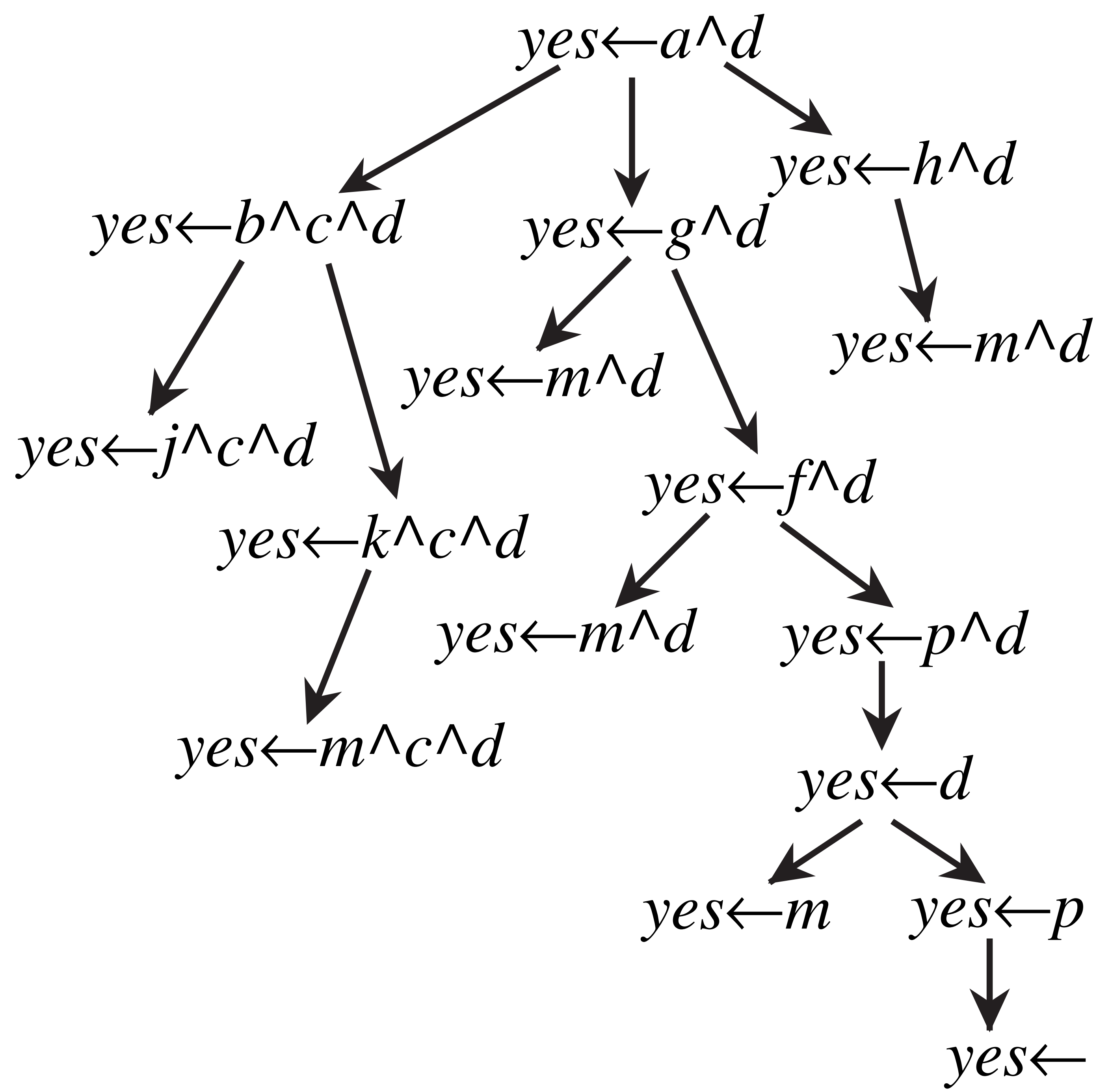
Example: failing derivation

$$
\begin{array}{lll}
a \leftarrow b \wedge c. & a \leftarrow e \wedge f. & b \leftarrow f \wedge k. \\
c \leftarrow e. & d \leftarrow k. & e. \\
f \leftarrow j \wedge e. & f \leftarrow c. & j \leftarrow c.
\end{array}
$$

Query: $?a$

$$
\begin{array}{llll}
\gamma_0 : & yes \leftarrow a & \gamma_4 : & yes \leftarrow e \wedge k \wedge c \\
\gamma_1 : & yes \leftarrow b \wedge c & \gamma_5 : & yes \leftarrow k \wedge c \\
\gamma_2 : & yes \leftarrow f \wedge k \wedge c & & \\
\gamma_3 : & yes \leftarrow c \wedge k \wedge c & &
\end{array}
$$

Search Graph for SLD Resolution



$a \leftarrow b \wedge c.$

$a \leftarrow h.$

$b \leftarrow k.$

$d \leftarrow p.$

$sf \leftarrow p.$

$g \leftarrow f.$

$h \leftarrow m.$

$?a \wedge d$

$a \leftarrow g.$

$b \leftarrow j.$

$d \leftarrow m.$

$f \leftarrow m.$

$g \leftarrow m.$

$k \leftarrow m.$

$p.$