Propositions

- An interpretation is an assignment of values to all variables.
- A model is an interpretation that satisfies the constraints.
- Often we don't want to just find a model, but want to know what is true in all models.
- A proposition is statement that is true or false in each interpretation.

Why Propositions?

- Specifying logical formulae is often more natural than filling in tables
- It is easier to check correctness and debug formulae than tables
- We can exploit the Boolean nature for efficient reasoning
- We need a language for asking queries (of what follows in all models) that may be more complicated than asking for the value of a variable
- It is easy to incrementally add formulae
- It can be extended to infinitely many variables with infinite domains (using logical quantification)

Representation and Reasoning System

A Representation and Reasoning System (RRS) is made up of:

- formal language: specifies the legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: nondeterministic specification of how an answer can be produced.

Using an RRS

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology).
3. Choose symbols in the computer to denote propositions
4. Tell the system knowledge about the domain.
5. Ask the system questions.
6. — the system can tell you whether your question is true

Role of semantics

In Computer:

$$l1\_broken \leftarrow sw\_up$$
$$\wedge power \wedge unlit\_l1.$$
$$sw\_up.$$
$$power \leftarrow lit\_l2.$$
$$unlit\_l1.$$
$$lit\_l2.$$

In user's mind:

- $l1\_broken$: light $l1$ is broken
- sw_up: switch is up
- $power$: there is power in the building
- $unlit\_l1$: light $l1$ isn't lit
- lit_l2: light $l2$ is lit

Conclusion: $l1\_broken$

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

Simple language: propositional definite clauses

- An **atom** is a symbol starting with a lower case letter
- A **body** is an atom or is of the form $b_1 \wedge b_2$ where b_1 and b_2 are bodies.
- A **definite clause** is an atom or is a rule of the form $h \leftarrow b$ where h is an atom and b is a body.
- A **knowledge base** is a set of definite clauses

Semantics

- An interpretation I assigns a truth value to each atom.
- A body $b_1 \wedge b_2$ is true in I if b_1 is true in I and b_2 is true in I.
- A rule $h \leftarrow b$ is false in I if b is true in I and h is false in I. The rule is true otherwise.
- A knowledge base KB is true in I if and only if every clause in KB is true in I.

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 clauses and g is a conjunction of atoms, 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

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	p	q	r	s	
I_1	*true*	*true*	*true*	*true*	is a model of KB
I_2	*false*	*false*	*false*	*false*	not a model of KB
I_3	*true*	*true*	*false*	*false*	is a model of KB
I_4	*true*	*true*	*true*	*false*	is a model of KB
I_5	*true*	*true*	*false*	*true*	not a model of KB

$KB \models p$, $KB \models q$, $KB \not\models r$, $KB \not\models s$

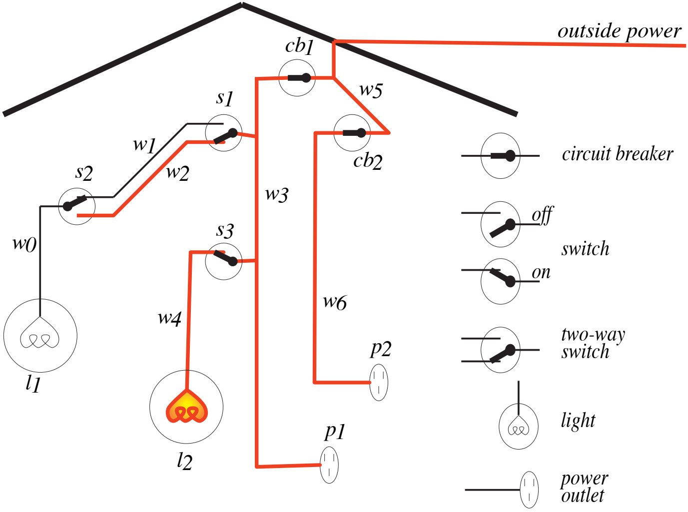
User's view of Semantics

1. Choose a task domain: intended interpretation.
2. Associate an atom with each proposition you want to represent.
3. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
4. Ask questions about the intended interpretation.
5. If $KB \models g$, then g must be true in the intended interpretation.
6. The use can interpret the answer using their intended interpretation of the symbols.

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



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 $KB \models g$.
- A proof procedure is complete if $KB \models g$ implies $KB \vdash g$.

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.*

This is forward chaining on this clause.
(This rule also covers the case when $m = 0$.)

Bottom-up proof procedure

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

$C := \{\}$;
repeat
    select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" 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.

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.
- 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.

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.**

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 Ground 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

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

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, and
  - 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

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

Query: $?a$

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

Example: failing derivation

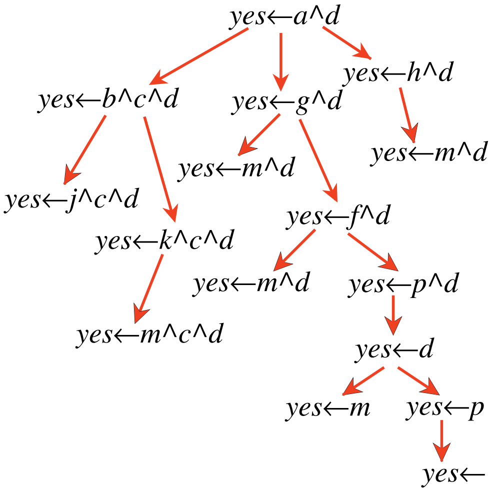
$$a \leftarrow b \wedge c. \qquad a \leftarrow e \wedge f. \qquad b \leftarrow f \wedge k.$$
$$c \leftarrow e. \qquad d \leftarrow k. \qquad e.$$
$$f \leftarrow j \wedge e. \qquad f \leftarrow c. \qquad j \leftarrow c.$$

Query: $?a$

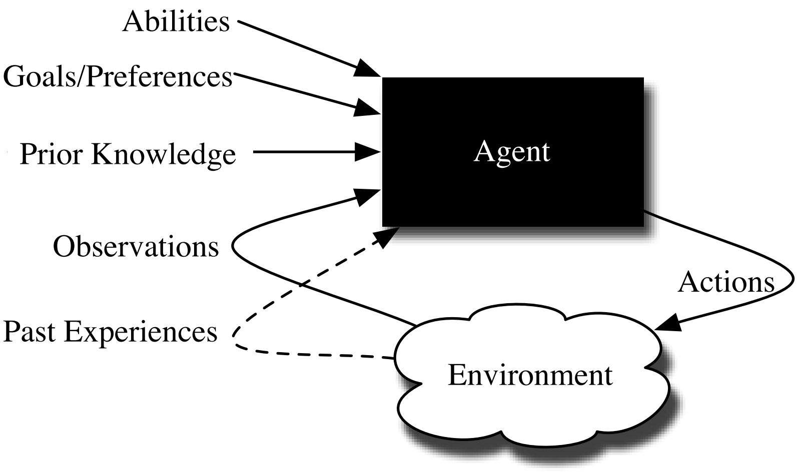
$$\gamma_0 : \quad yes \leftarrow a$$
$$\gamma_1 : \quad yes \leftarrow b \wedge c$$
$$\gamma_2 : \quad yes \leftarrow f \wedge k \wedge c$$
$$\gamma_3 : \quad yes \leftarrow c \wedge k \wedge c$$
$$\gamma_4 : \quad yes \leftarrow e \wedge k \wedge c$$
$$\gamma_5 : \quad yes \leftarrow k \wedge c$$

Search Graph for SLD Resolution

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



Agents acting in an environment



Users

- In the electrical domain, what should the house builder know?
- What should an occupant know?

Users

- In the electrical domain, what should the house builder know?
- What should an occupant know?
- Users can't be expected to volunteer knowledge:
  - They don't know what information is needed.
  - They don't know what vocabulary to use.

Ask-the-user

- Users can provide observations to the system. They can answer specific queries.
- Askable atoms are those that a user should be able to observe.
- There are 3 sorts of goals in the top-down proof procedure:
  - Goals for which the user isn't expected to know the answer.
  - Askable atoms that may be useful in the proof.
  - Askable atoms that the user has already provided information about.

Ask-the-user

- Users can provide observations to the system. They can answer specific queries.
- Askable atoms are those that a user should be able to observe.
- There are 3 sorts of goals in the top-down proof procedure:
  - Goals for which the user isn't expected to know the answer.
  - Askable atoms that may be useful in the proof.
  - Askable atoms that the user has already provided information about.
- The top-down proof procedure can be modified to ask users about askable atoms they have not already provided answers for.

Knowledge-Level Explanation

- HOW questions can be used to ask how an atom was proved.
  It gives the rule used to prove the atom.
  You can the ask HOW an element of the body of that rules was proved.
  This lets the user explore the proof.
- WHY questions can be used to ask why a question was asked.
  It provides the rule with the asked atom in the body.
  You can ask WHY the rule in the head was asked.

Knowledge-Level Debugging

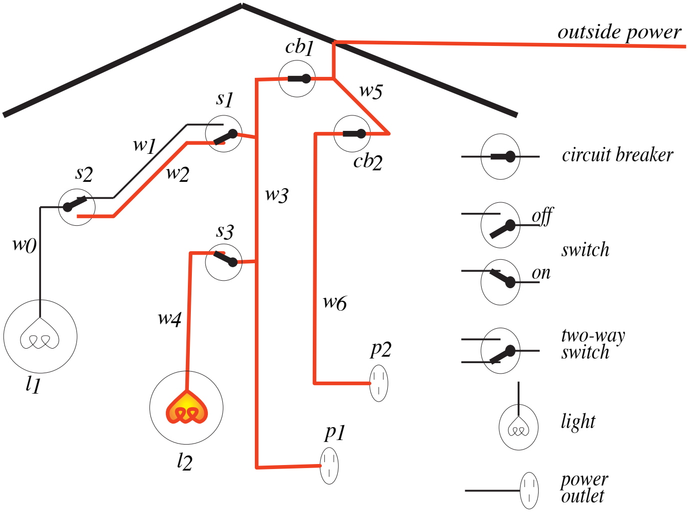
There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.
- Some answer wasn't produced: the proof failed when it should have succeeded. Some particular true atom wasn't derived.
- The program gets into an infinite loop.
- The system asks irrelevant questions.

Debugging incorrect answers

- Suppose atom g was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ in the knowledge base that was used to prove g.
- Either:
  - one of the a_i is false in the intended interpretation or
  - all of the a_i are true in the intended interpretation.
- Incorrect answers can be debugged by only answering yes/no questions.

Electrical Environment



Missing Answers

If atom g is true in the intended interpretation, but could not be proved, either:

- There is no appropriate rule for g.
- There is a rule $g \leftarrow a_1 \wedge \ldots \wedge a_k$ that should have succeeded.

Integrity Constraints

- In the electrical domain, what if we predict that a light should be on, but observe that it isn't?
  What can we conclude?
- We will expand the definite clause language to include integrity constraints which are rules that imply *false*, where *false* is an atom that is false in all interpretations.
- This will allow us to make conclusions from a contradiction.
- A definite clause knowledge base is always consistent.
  This won't be true with the rules that imply *false*.

Horn clauses

- An integrity constraint is a clause of the form

  $$false \leftarrow a_1 \wedge \ldots \wedge a_k$$

  where the a_i are atoms and *false* is a special atom that is false in all interpretations.
- A Horn clause is either a definite clause or an integrity constraint.

Negative Conclusions

- Negations can follow from a Horn clause KB.
- The negation of α, written $\neg\alpha$ is a formula that
  - is true in interpretation I if α is false in I, and
  - is false in interpretation I if α is true in I.
- Example:

$$KB = \left\{ \begin{array}{l} false \leftarrow a \wedge b. \\ a \leftarrow c. \\ b \leftarrow c. \end{array} \right\} \qquad KB \models \neg c.$$

Disjunctive Conclusions

- Disjunctions can follow from a Horn clause KB.
- The disjunction of α and β, written $\alpha \vee \beta$, is
  - true in interpretation I if α is true in I or β is true in I (or both are true in I).
  - false in interpretation I if α and β are both false in I.
- Example:

$$KB = \left\{ \begin{array}{l} false \leftarrow a \wedge b. \\ a \leftarrow c. \\ b \leftarrow d. \end{array} \right\} \qquad KB \models \neg c \vee \neg d.$$

Questions and Answers in Horn KBs

- An assumable is an atom whose negation you are prepared to accept as part of a (disjunctive) answer.
- A conflict of *KB* is a set of assumables that, given *KB* imply *false*.
- A minimal conflict is a conflict such that no strict subset is also a conflict.

Conflict Example

Example: If $\{c, d, e, f, g, h\}$ are the assumables

$$KB = \left\{ \begin{array}{l} false \leftarrow a \wedge b. \\ a \leftarrow c. \\ b \leftarrow d. \\ b \leftarrow e. \end{array} \right\}$$

- $\{c, d\}$ is a conflict
- $\{c, e\}$ is a conflict
- $\{c, d, e, h\}$ is a conflict

Using Conflicts for Diagnosis

- Assume that the user is able to observe whether a light is lit or dark and whether a power outlet is dead or live.
- A light can't be both lit and dark. An outlet can't be both live and dead:

$$false \leftarrow dark\_l_1 \;\&\; lit\_l_1.$$

$$false \leftarrow dark\_l_2 \;\&\; lit\_l_2.$$

$$false \leftarrow dead\_p_1 \;\&\; live\_p_2.$$

- Assume the individual components are working correctly:

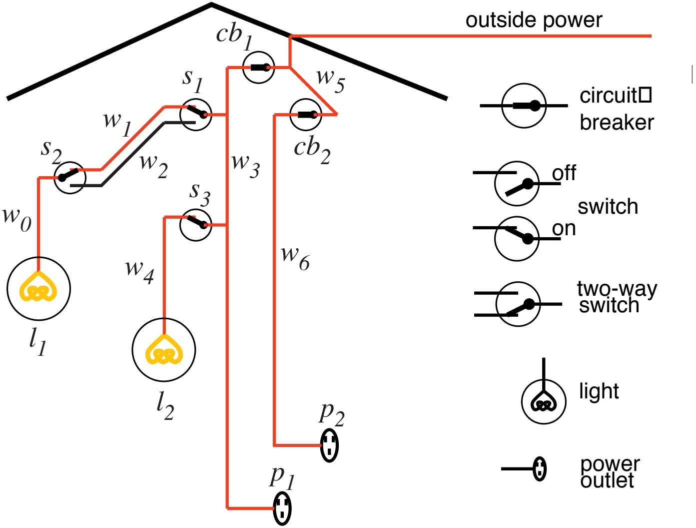
$$assumable\ ok\_l_1.$$

$$assumable\ ok\_s_2.$$

$$\ldots$$

- Suppose switches s_1, s_2, and s_3 are all up:
$up\_s_1.\ up\_s_2.\ up\_s_3.$

Electrical Environment



Representing the Electrical Environment

$light\_l_1.$

$light\_l_2.$

$up\_s_1.$

$up\_s_2.$

$up\_s_3.$

$live\_outside.$

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

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

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

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

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

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

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

$live\_p_1 \leftarrow live\_w_3.$

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

$live\_p_2 \leftarrow live\_w_6.$

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

$live\_w_5 \leftarrow live\_outside.$

- If the user has observed l_1 and l_2 are both dark:

  $$dark\_l_1.\ dark\_l_2.$$

- There are two minimal conflicts:

  $\{ok\_cb_1, ok\_s_1, ok\_s_2, ok\_l_1\}$ and

  $\{ok\_cb_1, ok\_s_3, ok\_l_2\}$.

- You can derive:

  $\neg ok\_cb_1 \vee \neg ok\_s_1 \vee \neg ok\_s_2 \vee \neg ok\_l_1$

  $\neg ok\_cb_1 \vee \neg ok\_s_3 \vee \neg ok\_l_2$.

- Either cb_1 is broken or there is one of six double faults.

Diagnoses

- A **consistency-based diagnosis** is a set of assumables that has at least one element in each conflict.
- A **minimal diagnosis** is a diagnosis such that no subset is also a diagnosis.
- Intuitively, one of the minimal diagnoses must hold. A diagnosis holds if all of its elements are false.
- Example: For the proceeding example there are seven minimal diagnoses: $\{ok\_cb_1\}$, $\{ok\_s_1, ok\_s_3\}$, $\{ok\_s_1, ok\_l_2\}$, $\{ok\_s_2, ok\_s_3\}, \ldots$

Recall: top-down consequence finding

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.

Implementing conflict finding: top down

- Query is *false*.
- Don't select an atom that is assumable.
- Stop when all of the atoms in the body of the generalised query are assumable:
    - this is a conflict

Example

$false \leftarrow a.$

$a \leftarrow b \,\&\, c.$

$b \leftarrow d.$

$b \leftarrow e.$

$c \leftarrow f.$

$c \leftarrow g.$

$e \leftarrow h \,\&\, w.$

$e \leftarrow g.$

$w \leftarrow f.$

assumable $d, f, g, h.$

Bottom-up Conflict Finding

- Conclusions are pairs $\langle a, A \rangle$, where a is an atom and A is a set of assumables that imply a.
- Initially, conclusion set $C = \{\langle a, \{a\} \rangle : a \text{ is assumable}\}$.
- If there is a rule $h \leftarrow b_1 \wedge \ldots \wedge b_m$ such that for each b_i there is some A_i such that $\langle b_i, A_i \rangle \in C$, then $\langle h, A_1 \cup \ldots \cup A_m \rangle$ can be added to C.
- If $\langle a, A_1 \rangle$ and $\langle a, A_2 \rangle$ are in C, where $A_1 \subset A_2$, then $\langle a, A_2 \rangle$ can be removed from C.
- If $\langle \mathit{false}, A_1 \rangle$ and $\langle a, A_2 \rangle$ are in C, where $A_1 \subseteq A_2$, then $\langle a, A_2 \rangle$ can be removed from C.

Bottom-up Conflict Finding Code

$C := \{\langle a, \{a\}\rangle : a \text{ is assumable } \};$
repeat
    select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" in T such that
        $\langle b_i, A_i\rangle \in C$ for all i and
        there is no $\langle h, A'\rangle \in C$ or $\langle \mathit{false}, A'\rangle \in C$
            such that $A' \subseteq A$ where $A = A_1 \cup \ldots \cup A_m$;
    $C := C \cup \{\langle h, A\rangle\}$
    Remove any elements of C that can now be pruned;
until no more selections are possible

Complete Knowledge Assumption

- Often you want to assume that your knowledge is complete.
- Example: you can state what switches are up and the agent can assume that the other switches are down.
- Example: assume that a database of what students are enrolled in a course is complete.
- The definite clause language is monotonic: adding clauses can't invalidate a previous conclusion.
- Under the complete knowledge assumption, the system is non-monotonic: adding clauses can invalidate a previous conclusion.

Completion of a knowledge base

- Suppose the rules for atom a are

$$a \leftarrow b_1.$$
$$\vdots$$
$$a \leftarrow b_n.$$

  equivalently $a \leftarrow b_1 \vee \ldots \vee b_n.$

- Under the Complete Knowledge Assumption, if a is true, one of the b_i must be true:

$$a \rightarrow b_1 \vee \ldots \vee b_n.$$

- Under the CKA, the clauses for a mean Clark's completion:

$$a \leftrightarrow b_1 \vee \ldots \vee b_n$$

Clark's Completion of a KB

- Clark's completion of a knowledge base consists of the completion of every atom.
- If you have an atom a with no clauses, the completion is $a \leftrightarrow false$.
- You can interpret negations in the body of clauses.
  $\sim a$ means that a is false under the complete knowledge assumption
  This is negation as failure.

Bottom-up negation as failure interpreter

$C := \{\};$
repeat
  either
    select $r \in KB$ such that
      r is "$h \leftarrow b_1 \wedge \ldots \wedge b_m$"
      $b_i \in C$ for all i, and
      $h \notin C$;
    $C := C \cup \{h\}$
  or
    select h such that for every rule "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" $\in KB$
      either for some $b_i, \sim b_i \in C$
      or some $b_i = \sim g$ and $g \in C$
    $C := C \cup \{\sim h\}$
until no more selections are possible

Negation as failure example

$$p \leftarrow q \land \sim r.$$

$$p \leftarrow s.$$

$$q \leftarrow \sim s.$$

$$r \leftarrow \sim t.$$

$$t.$$

$$s \leftarrow w.$$

Top-Down negation as failure proof procedure

- If the proof for a fails, you can conclude $\sim a$.
- Failure can be defined recursively:
  Suppose you have rules for atom a:

$$a \leftarrow b_1$$
$$\vdots$$
$$a \leftarrow b_n$$

  If each body b_i fails, a fails.
  A body fails if one of the conjuncts in the body fails.
  Note that you need *finite* failure. Example $p \leftarrow p$.

Assumption-based Reasoning

Often we want our agents to make assumptions rather than doing deduction from their knowledge. For example:

- In abduction an agent makes assumptions to explain observations. For example, it hypothesizes what could be wrong with a system to produce the observed symptoms.
- In default reasoning an agent makes assumptions of normality to make predictions. For example, the delivery robot may want to assume Mary is in her office, even if it isn't always true.

Design and Recognition

Two different tasks use assumption-based reasoning:

- Design The aim is to design an artifact or plan. The designer can select whichever design they like that satisfies the design criteria.
- Recognition The aim is to find out what is true based on observations. If there are a number of possibilities, the recognizer can't select the one they like best. The underlying reality is fixed; the aim is to find out what it is.

Compare: Recognizing a disease with designing a treatment. Designing a meeting time with determining when it is.

The Assumption-based Framework

The assumption-based framework is defined in terms of two sets of formulae:

- F is a set of closed formula called the facts.
  These are formulae that are given as true in the world.
  We assume F are Horn clauses.
- H is a set of formulae called the possible hypotheses or assumables. Ground instance of the possible hypotheses can be assumed if consistent.

Making Assumptions

- A scenario of $\langle F, H\rangle$ is a set D of ground instances of elements of H such that $F \cup D$ is satisfiable.
- An explanation of g from $\langle F, H\rangle$ is a scenario that, together with F, implies g.
  D is an explanation of g if $F \cup D \models g$ and $F \cup D \not\models \mathit{false}$.
  A minimal explanation is an explanation such that no strict subset is also an explanation.
- An extension of $\langle F, H\rangle$ is the set of logical consequences of F and a maximal scenario of $\langle F, H\rangle$.

Example

$a \leftarrow b \land c.$
$b \leftarrow e.$
$b \leftarrow h.$
$c \leftarrow g.$
$c \leftarrow f.$
$d \leftarrow g.$
$false \leftarrow e \land d.$
$f \leftarrow h \land m.$
assumable $e, h, g, m, n.$

- $\{e, m, n\}$ is a scenario.
- $\{e, g, m\}$ is not a scenario.
- $\{h, m\}$ is an explanation for a.
- $\{e, h, m\}$ is an explanation for a.
- $\{e, g, h, m\}$ isn't an explanation.
- $\{e, h, m, n\}$ is a maximal scenario.
- $\{h, g, m, n\}$ is a maximal scenario.

Default Reasoning and Abduction

There are two strategies for using the assumption-based framework:

- **Default reasoning** Where the truth of g is unknown and is to be determined.
  An explanation for g corresponds to an **argument** for g.
- **Abduction** Where g is given, and we are interested in explaining it. g could be an observation in a recognition task or a design goal in a design task.

Default Reasoning

- When giving information, you don't want to enumerate all of the exceptions, even if you could think of them all.
- In default reasoning, you specify general knowledge and modularly add exceptions. The general knowledge is used for cases you don't know are exceptional.
- Classical logic is monotonic: If g logically follows from A, it also follows from any superset of A.
- Default reasoning is nonmonotonic: When you add that something is exceptional, you can't conclude what you could before.

Defaults as Assumptions

Default reasoning can be modeled using

- H is normality assumptions
- F states what follows from the assumptions

An explanation of g gives an argument for g.

Default Example

A reader of newsgroups may have a default:
"Articles about AI are generally interesting".

$$H = \{int\_ai(X)\},$$

where $int\_ai(X)$ means X is interesting if it is about AI.
With facts:

$$interesting(X) \leftarrow about\_ai(X) \wedge int\_ai(X).$$
$$about\_ai(art\_23).$$

$\{int\_ai(art\_23)\}$ is an explanation for $interesting(art\_23)$.

Default Example, Continued

We can have exceptions to defaults:

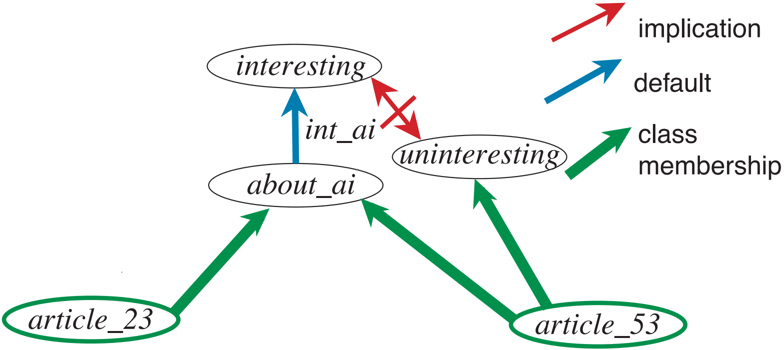
$$false \leftarrow interesting(X) \wedge uninteresting(X).$$

Suppose article 53 is about AI but is uninteresting:

$$about\_ai(art\_53).$$
$$uninteresting(art\_53).$$

We cannot explain $interesting(art\_53)$ even though everything we know about art_23 you also know about art_53.

Exceptions to defaults



Exceptions to Defaults

"Articles about formal logic are about AI."
"Articles about formal logic are uninteresting."
"Articles about machine learning are about AI."

$$about\_ai(X) \leftarrow about\_fl(X).$$

$$uninteresting(X) \leftarrow about\_fl(X).$$

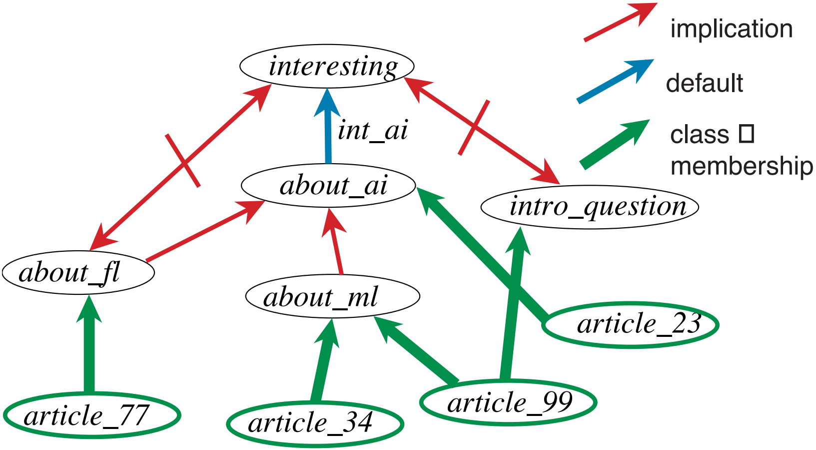
$$about\_ai(X) \leftarrow about\_ml(X).$$

$$about\_fl(art\_77).$$

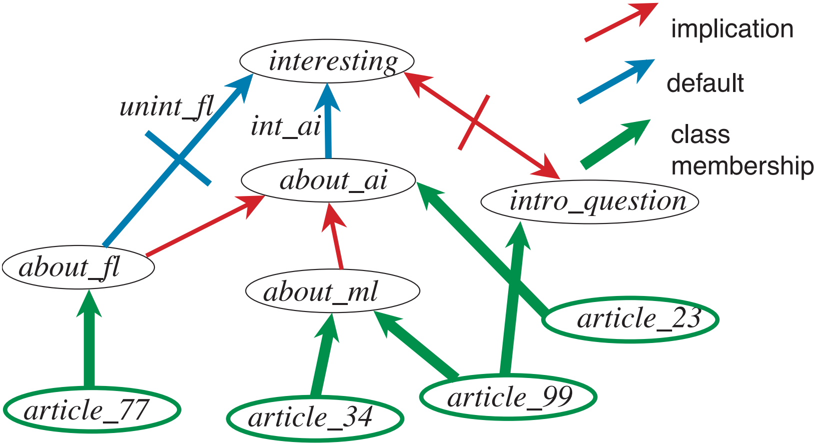
$$about\_ml(art\_34).$$

You can't explain $interesting(art\_77)$.
You can explain $interesting(art\_34)$.

Exceptions to Defaults



Formal logic is uninteresting by default



Contradictory Explanations

Suppose formal logic articles aren't interesting *by default*:

$$H = \{unint\_fl(X), int\_ai(X)\}$$

The corresponding facts are:

$$interesting(X) \leftarrow about\_ai(X) \land int\_ai(X).$$
$$about\_ai(X) \leftarrow about\_fl(X).$$
$$uninteresting(X) \leftarrow about\_fl(X) \land unint\_fl(X).$$
$$about\_fl(art\_77).$$

$uninteresting(art\_77)$ has explanation $\{unint\_fl(art\_77)\}$.
$interesting(art\_77)$ has explanation $\{int\_ai(art\_77)\}$.

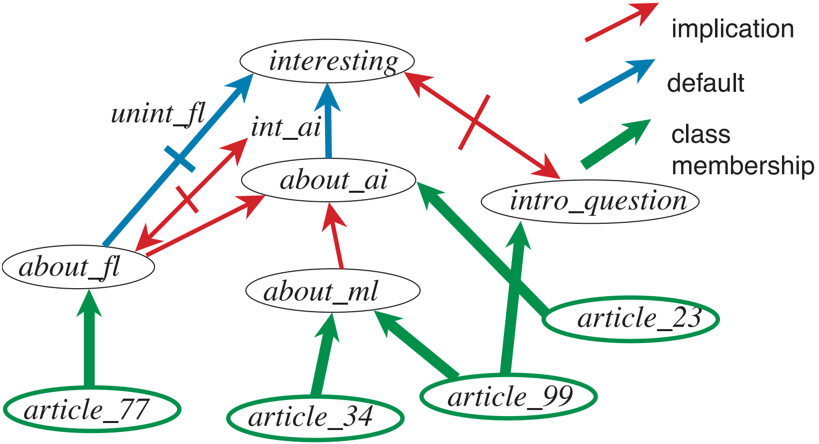
Overriding Assumptions

- Because art_77 is about formal logic, the argument "art_77 is interesting because it is about AI" shouldn't be applicable.
- This is an instance of preference for more specific defaults.
- Arguments that articles about formal logic are interesting because they are about AI can be defeated by adding:

  $$false \leftarrow \; about\_fl(X) \wedge int\_ai(X).$$

  This is known as a cancellation rule.
- You can no longer explain $interesting(art\_77)$.

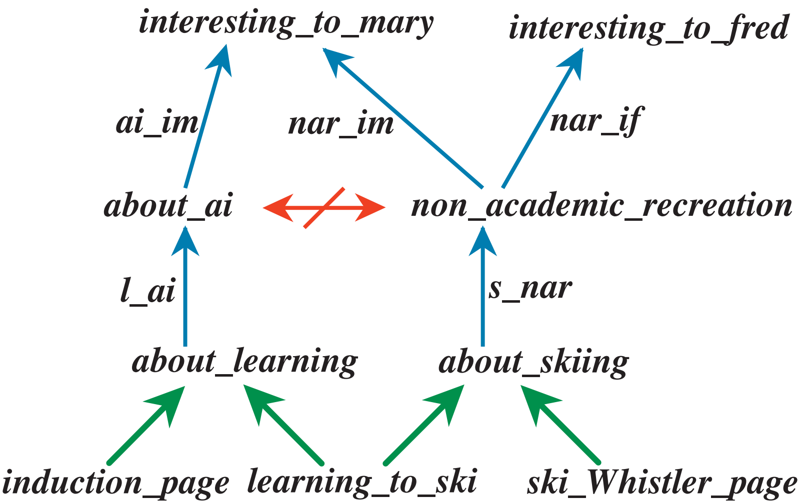
Diagram of the Default Example



Multiple Extension Problem

- What if incompatible goals can be explained and there are no cancellation rules applicable?
  What should we predict?
- For example: what if introductory questions are uninteresting, by default?
- This is the multiple extension problem.
- Recall: an extension of $\langle F, H\rangle$ is the set of logical consequences of F and a maximal scenario of $\langle F, H\rangle$.

Competing Arguments



Skeptical Default Prediction

- We predict g if g is in all extensions of $\langle F, H \rangle$.
- Suppose g isn't in extension E. As far as we are concerned E could be the correct view of the world. So we shouldn't predict g.
- If g is in all extensions, then no matter which extension turns out to be true, we still have g true.
- Thus g is predicted even if an adversary gets to select assumptions, as long as the adversary is forced to select something. You do not predict g if the adversary can pick assumptions from which g can't be explained.

Minimal Models Semantics for Prediction

Recall: logical consequence is defined as truth in all models.
We can define default prediction as truth in all **minimal models**.
Suppose M_1 and M_2 are models of the facts.
$M_1 <_H M_2$ if the hypotheses violated by M_1 are a strict subset of the hypotheses violated by M_2. That is:

$$\{h \in H' : h \text{ is false in } M_1\} \subset \{h \in H' : h \text{ is false in } M_2\}$$

where H' is the set of ground instances of elements of H.

Minimal Models and Minimal Entailment

- M is a **minimal model** of F with respect to H if M is a model of F and there is no model M_1 of F such that $M_1 <_H M$.
- g is **minimally entailed** from $\langle F, H \rangle$ if g is true in all minimal models of F with respect to H.
- **Theorem:** g is minimally entailed from $\langle F, H \rangle$ if and only if g is in all extensions of $\langle F, H \rangle$.

Evidential and Causal Reasoning

Much reasoning in AI can be seen as evidential reasoning, (observations to a theory) followed by causal reasoning (theory to predictions).

- Diagnosis Given symptoms, evidential reasoning leads to hypotheses about diseases or faults, these lead via causal reasoning to predictions that can be tested.
- Robotics Given perception, evidential reasoning can lead us to hypothesize what is in the world, that leads via causal reasoning to actions that can be executed.

Combining Evidential & Causal Reasoning

To combine evidential and causal reasoning, you can either

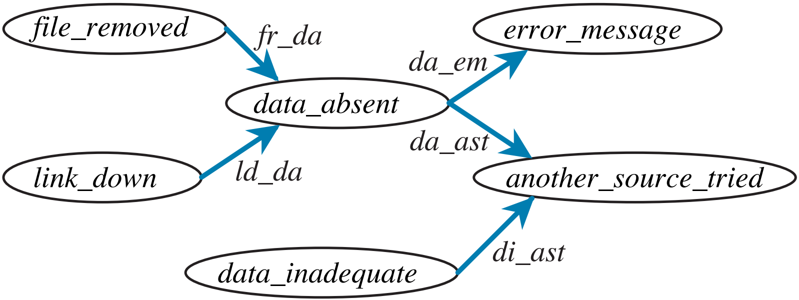
- Axiomatize from causes to their effects and
  - use abduction for evidential reasoning
  - use default reasoning for causal reasoning
- Axiomatize both
  - effects $\longrightarrow$ possible causes (for evidential reasoning)
  - causes $\longrightarrow$ effects (for causal reasoning)

  use a single reasoning mechanism, such as default reasoning.

Combining abduction and default reasoning

- Representation:
  - Axiomatize causally using rules.
  - Have normality assumptions (defaults) for prediction
  - other assumptions to explain observations
- Reasoning:
  - given an observation, use all assumptions to explain observation (find base causes)
  - use normality assumptions to predict from base causes explanations.

Causal Network



Why is the infobot trying another information source?
(Arrows are implications or defaults. Sources are assumable.)

Code for causal network

$$error\_message \leftarrow data\_absent \wedge da\_em.$$

$$another\_source\_tried \leftarrow data\_absent \wedge da\_ast$$

$$another\_source\_tried \leftarrow data\_inadequate \wedge di\_ast.$$

$$data\_absent \leftarrow file\_removed \wedge fr\_da.$$

$$data\_absent \leftarrow link\_down \wedge ld\_da.$$

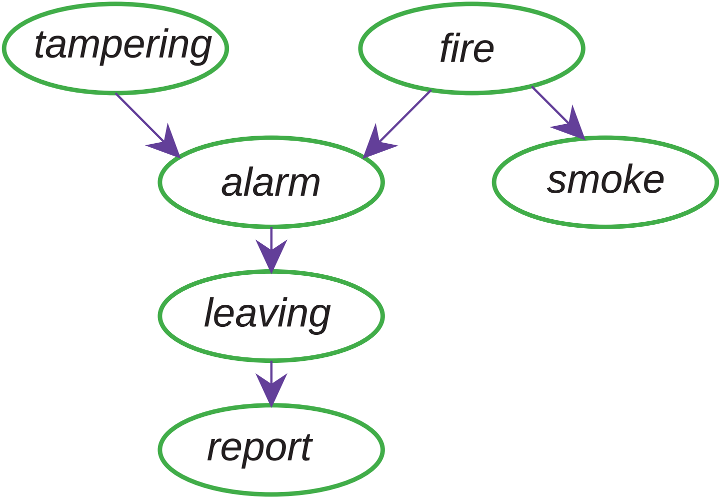
default $da\_em, da\_ast, di\_ast, fr\_da, ld\_da.$

assumable $file\_removed.$

assumable $link\_down.$

assumable $data\_inadequate.$

Example: fire alarm



Fire Alarm Code

assumable $tampering$.

assumable $fire$.

$alarm \leftarrow tampering \wedge tampering\_caused\_alarm$.

$alarm \leftarrow fire \wedge fire\_caused\_alarm$.

default $tampering\_caused\_alarm$.

default $fire\_caused\_alarm$.

$smoke \leftarrow fire \wedge fire\_caused\_smoke$.

default $fire\_caused\_smoke$.

$leaving \leftarrow alarm \wedge alarm\_caused\_leaving$.

default $alarm\_caused\_leaving$.

$report \leftarrow leaving \wedge leaving\_caused\_report$.

default $leaving\_caused\_report$.

Explaining Away

- If we observe *report* there are two minimal explanations:
  - one with *tampering*
  - one with *fire*
- If we observed just *smoke* there is one explanation (containing *fire*). This explanation makes no predictions about tampering.
- If we had observed *report* $\wedge$ *smoke*, there is one minimal explanation, (containing *fire*).
  - The smoke explains away the tampering. There is no need to hypothesise *tampering* to explain report.