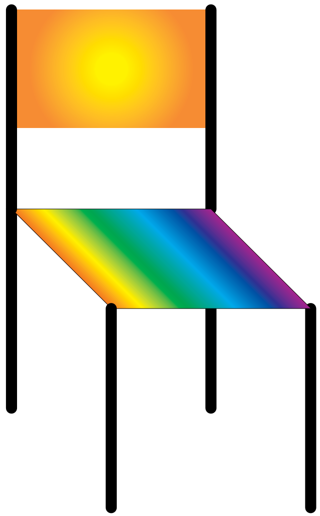
Chapter 7: Beyond Definite Knowledge

- Lecture 1 Equality, inequality and the unique names assumption
- Lecture 2 Complete knowledge assumption and negation as failure.
- Lecture 3 Integrity Constraints, consistency-based diagnosis.

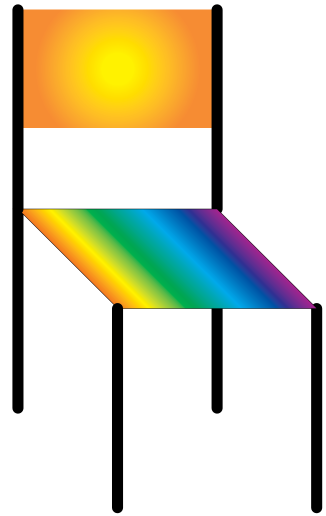
Equality

- Sometimes two terms denote the same individual.
- Example: Clark Kent & superman. 4×4 & $11 + 5$. The projector we used last Friday & this projector.
- Ground term t_1 equals ground term t_2, written $t_1 = t_2$, is true in interpretation I if t_1 and t_2 denote the same individual in interpretation I.

Equality doesn't mean similarity



chair 1



chair 2

$chair1 \neq chair2$

$chair\_on\_right = chair2$

$chair\_on\_right$ is not similar to $chair2$, it is $chair2$.

Why is equality important?

- In a doctor's office, the doctor wants to know if a patient is the same patient that she saw last week (or is his twin sister).
- In a criminal investigation, the police want to determine if someone is the same person as the person who committed some crime.
- When buying a replacement switch, an electrician may want to know if it was built in the same factory as the switches that were unreliable. (And if it is a different switch to the one that was replaced the previous time).

Allowing Equality Assertions

- Without equality assertions, the only thing that is equal to a ground term is itself.

  This can be captured as though you had the assertion $X = X$. Explicit equality never needs to be used.

- If you allow equality assertions, you need to derive what follows from them. Either:
  - axiomatize equality like any other predicate
  - build special-purpose inference machinery for equality

Axiomatizing Equality

$$X = X.$$

$$X = Y \leftarrow Y = X.$$

$$X = Z \leftarrow X = Y \wedge Y = Z.$$

For each n-ary function symbol f there is a rule of the form

$$f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \leftarrow$$
$$X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n.$$

For each n-ary predicate symbol p, there is a rule of the form

$$p(X_1, \ldots, X_n) \leftarrow$$
$$p(Y_1, \ldots, Y_n) \wedge X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n.$$

Special-Purpose Equality Reasoning

paramodulation: if you have $t_1 = t_2$, then you can replace any occurrence of t_1 by t_2.

Treat equality as a rewrite rule, substituting equals for equals.

You select a canonical representation for each individual and rewrite all other representations into that representation.

Example: treat the sequence of digits as the canonical representation of the number.

Example: use the student number as the canonical representation for students.

Unique Names Assumption

The convention that different ground terms denote different individuals is the **unique names assumption.**

For every pair of distinct ground terms t_1 and t_2, assume $t_1 \neq t_2$, where "$\neq$" means "not equal to."

Example: For each pair of courses, you don't want to have to state, $math302 \neq psyc303$, …

Example: Sometimes the unique names assumption is inappropriate, for example $3 + 7 \neq 2 \times 5$ is wrong.

Axiomatizing Inequality for the UNA

- $c \neq c'$ for any distinct constants c and c'.
- $f(X_1, \ldots, X_n) \neq g(Y_1, \ldots, Y_m)$ for any distinct function symbols f and g.
- $f(X_1, \ldots, X_n) \neq f(Y_1, \ldots, Y_n) \leftarrow X_i \neq Y_i$, for any function symbol f. There are n instances of this schema for every n-ary function symbol f (one for each i such that $1 \leq i \leq n$).
- $f(X_1, \ldots, X_n) \neq c$ for any function symbol f and constant c.
- $t \neq X$ for any term t in which X appears (where t is not the term X).

Top-down procedure and the UNA

- Inequality isn't just another predicate. There are infinitely many answers to $X \neq f(Y)$.

- If you have a subgoal $t_1 \neq t_2$, for terms t_1 and t_2 there are three cases:

  - t_1 and t_2 don't unify. In this case, $t_1 \neq t_2$ succeeds.

  - t_1 and t_2 are identical including having the same variables in the same positions. Here $t_1 \neq t_2$ fails.

  - Otherwise, there are instances of $t_1 \neq t_2$ that succeed and instances of $t_1 \neq t_2$ that fail.

Implementing the UNA

- Recall: in SLD resolution you can select any subgoal in the body of an answer clause to solve next.
- Idea: only select inequality when it will either succeed or fail, otherwise select another subgoal. Thus you are delaying inequality goals.
- If only inequality subgoals remain, and none fail, the query succeeds.

Inequality Example

$$notin(X, []).$$

$$notin(X, [H|T]) \leftarrow X \neq H \wedge notin(X, T).$$

$$good\_course(C) \leftarrow course(C) \wedge passes\_analysis(C).$$

$$course(cs312).$$

$$course(cs444).$$

$$course(cs322).$$

$$passes\_analysis(C) \leftarrow something\_complicated(C).$$

$$?notin(C, [cs312, cs322, cs422, cs310, cs402])$$

$$\quad \wedge good\_course(C).$$

Complete Knowledge Assumption (CKA)

Sometimes you want to assume that a database of facts is complete. Any fact not listed is false.

Example: Assume that a database of *enrolled* relations is complete. Then you can define *empty_course*.

Example: Assume a database of video segments is complete.

The definite clause RRS is monotonic: adding clauses doesn't invalidate a previous conclusion.

With the complete knowledge assumption, the system is nonmonotonic: a conclusion can be invalidated by adding more clauses (but this must not be allowed).

CKA: propositional case

Suppose the rules for atom a are

$$a \leftarrow b_1.$$

$$\ldots$$

$$a \leftarrow b_n.$$

or equivalently: $a \leftarrow b_1 \vee \ldots \vee b_n$

Under the CKA, 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:

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

CKA: Ground Database

Example: Consider the relation defined by:

$$student(mary).$$

$$student(john).$$

$$student(ying).$$

The CKA specifies these three are the only students:

$$student(X) \leftrightarrow X = mary \vee X = john \vee X = ying.$$

To conclude $\neg student(alan)$, you have to be able to prove

$$alan \neq mary \wedge alan \neq john \wedge alan \neq ying$$

This needs the unique names assumption.

Clark Normal Form

The Clark normal form of the clause:

$$p(t_1, \ldots, t_k) \leftarrow B$$

is the clause

$$p(V_1, \ldots, V_k) \leftarrow \exists W_1 \ldots \exists W_m \; V_1 = t_1 \wedge \ \ldots \wedge V_k = t_k \wedge B,$$

where $V_1, \ldots, V_k$ are k different variables that did not appear in the original clause.

$W_1, \ldots, W_m$ are the original variables in the clause.

Clark normal form: example

- The Clark normal form of:

$$room(C, room208) \leftarrow$$
$$cs\_course(C) \wedge enrollment(C, E) \wedge E < 120.$$

is

$$room(X, Y) \leftarrow \exists C \exists E\ X = C \wedge Y = room208 \wedge$$
$$cs\_course(C) \wedge enrollment(C, E) \wedge E < 120.$$

Clark's Completion of a Predicate

Put all of the clauses for p into Clark normal form, with the same set of introduced variables:

$$p(V_1, \ldots, V_k) \leftarrow B_1$$
$$\vdots$$
$$p(V_1, \ldots, V_k) \leftarrow B_n$$

This is the same as: $p(V_1, \ldots, V_k) \leftarrow B_1 \vee \ldots \vee B_n.$

Clark's completion of p is the equivalence

$$p(V_1, \ldots, V_k) \leftrightarrow B_1 \vee \ldots \vee B_n,$$

That is, $p(V_1, \ldots, V_k)$ is true if and only if one B_i is true.

Clark's Completion Example

Given the *mem* function:

$$mem(X, [X|T]).$$

$$mem(X, [H|T]) \leftarrow mem(X, T).$$

the completion is

$$mem(X, Y) \iff (\exists T\ Y = [X|T]) \vee$$
$$(\exists H \exists T\ Y = [H|T] \wedge mem(X, T))$$

Clark's Completion of a KB

- **Clark's completion** of a knowledge base consists of the completion of every predicate symbol, along with the axioms for equality and inequality.
- If you have a predicate p defined by no clauses in the knowledge base, the completion is $p \leftrightarrow false$. That is, $\neg p$.
- You can interpret negations in the bodies of clauses. $\sim p$ means that p is false under the Complete Knowledge Assumption. This is called **negation as failure.**

Using negation as failure

Previously we couldn't define $empty\_course(C)$ from a database of $enrolled(S, C)$.

This can be defined using negation as failure:

$$
\begin{aligned}
& empty\_course(C) \leftarrow \\
& \qquad course(C) \land \\
& \qquad \sim has\_Enrollment(C). \\
& has\_Enrollment(C) \leftarrow \\
& \qquad enrolled(S, C).
\end{aligned}
$$

Bottom-up NAF proof procedure

$C := \{\}$;

repeat

either select "$h \leftarrow b_1 \wedge \ldots \wedge b_m$"$\in KB$ such that

$b_i \in C$ for all i, and $h \notin C$;

$C := C \cup \{h\}$

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

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

$p \leftarrow s.$

$q \leftarrow \sim s.$

$r \leftarrow \sim t.$

$t.$

$s \leftarrow w.$

Top-Down NAF 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 require *finite* failure. Example: $p \leftarrow p$.

Free Variables in Negation as Failure

Example:

$$p(X) \leftarrow \sim q(X) \wedge r(X).$$

$$q(a).$$

$$q(b).$$

$$r(d).$$

There is only one answer to the query $?p(X)$, namely $X = d$.

For calls to negation as failure with free variables, you need to delay negation as failure goals that contain free variables until the variables become bound.

Floundering Goals

If the variables never become bound, a negated goal **flounders.**

In this case you can't conclude anything about the goal.

Example: Consider the clauses:

$$p(X) \leftarrow \sim q(X)$$

$$q(X) \leftarrow \sim r(X)$$

$$r(a)$$

and the query

$$?p(X).$$

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:

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

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:

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

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:

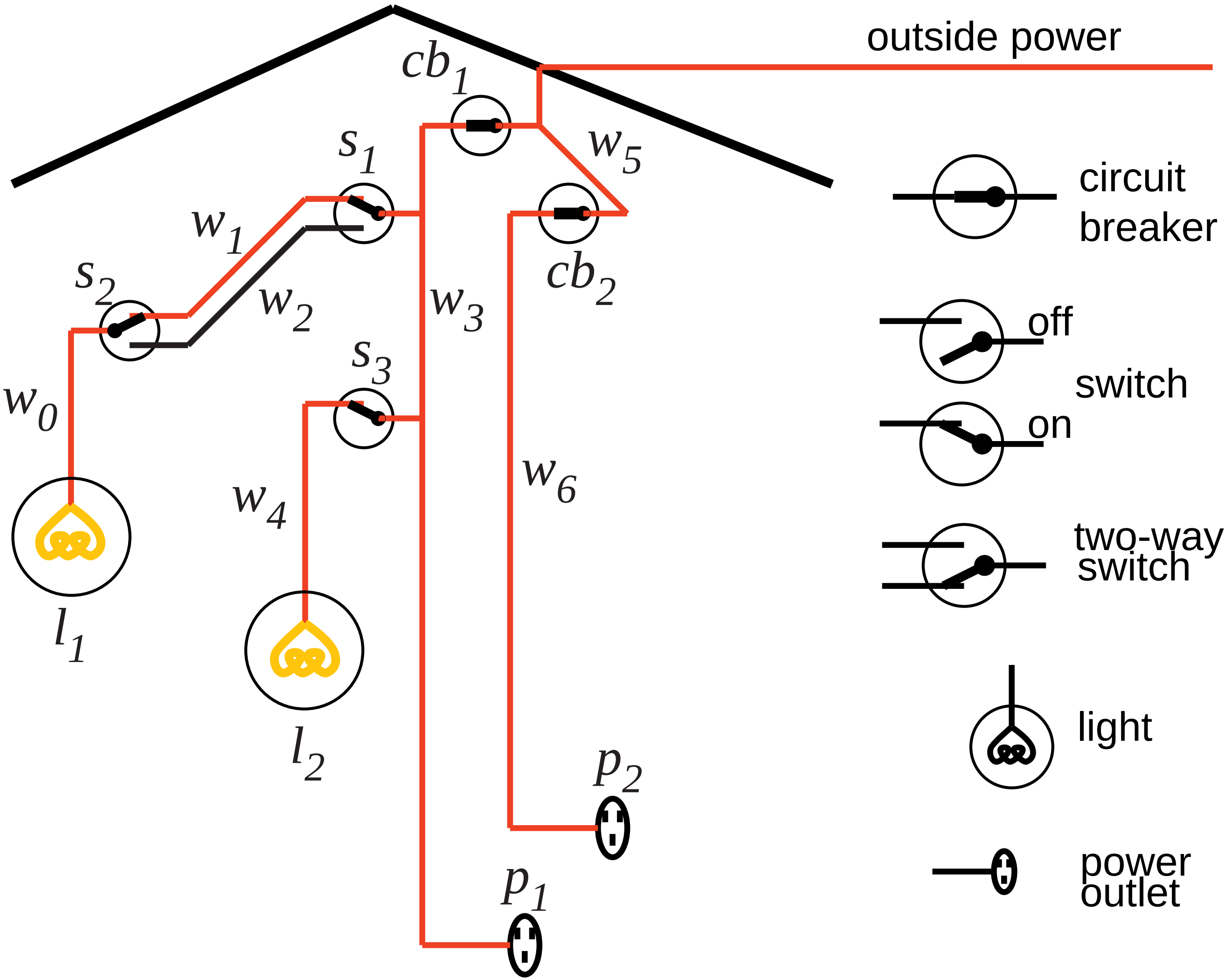
$$false \Leftarrow dark(L)\ \&\ lit(L).$$

$$false \Leftarrow dead(L)\ \&\ live(L).$$

- Make *ok* assumable: $assumable(ok(X))$.

- Suppose switches s_1, s_2, and s_3 are all up:
$up(s_1)$. $up(s_2)$. $up(s_3)$.

Electrical Environment



$$lit(L) \Leftarrow light(L) \,\&\, ok(L) \,\&\, live(L).$$

$$live(W) \Leftarrow connected\_to(W, W_1) \,\&\, live(W_1).$$

$$live(outside) \Leftarrow true.$$

$$light(l_1) \Leftarrow true.$$

$$light(l_2) \Leftarrow true.$$

$$connected\_to(l_1, w_0) \Leftarrow true.$$

$$connected\_to(w_0, w_1) \Leftarrow up(s_2) \,\&\, ok(s_2).$$

$$connected\_to(w_1, w_3) \Leftarrow up(s_1) \,\&\, ok(s_1).$$

$$connected\_to(w_3, w_5) \Leftarrow ok(cb_1).$$

$$connected\_to(w_5, outside) \Leftarrow true.$$

- 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: $\{ok(cb_1)\}$, $\{ok(s_1), ok(s_3)\}$, $\{ok(s_1), ok(l_2)\}$, $\{ok(s_2), ok(s_3)\}$,...

Meta-interpreter to find conflicts

% $dprove(G, D_0, D_1)$ is true if list D_0 is an ending of list D_1

% such that assuming the elements of D_1 lets you derive G.

$$dprove(true, D, D).$$

$$dprove((A \,\&\, B), D_1, D_3) \leftarrow$$

$$\quad dprove(A, D_1, D_2) \wedge dprove(B, D_2, D_3).$$

$$dprove(G, D, [G|D]) \leftarrow assumable(G).$$

$$dprove(H, D_1, D_2) \leftarrow$$

$$\quad (H \Leftarrow B) \wedge dprove(B, D_1, D_2).$$

$$conflict(C) \leftarrow dprove(false, [\,], C).$$

Tricky Example

$false \Leftarrow a.$

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

$b \Leftarrow d.$

$b \Leftarrow e.$

$c \Leftarrow f.$

$c \Leftarrow g.$

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

$e \Leftarrow g.$

$w \Leftarrow d.$

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

$C := \{\langle a, \{a\}\rangle : a$ 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

Integrity Constraints in Databases

- Database designers can use integrity constraints to specify constraints that should never be violated.

- Example: A student can't have two different grades for the same course.

$$
\begin{aligned}
& \mathit{false} \leftarrow \\
& \qquad \mathit{grade}(St, Course, Gr_1) \wedge \\
& \qquad \mathit{grade}(St, Course, Gr_2) \wedge \\
& \qquad Gr_1 \neq Gr_2.
\end{aligned}
$$

- When false is derived, HOW can be used to debug the KB.