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:

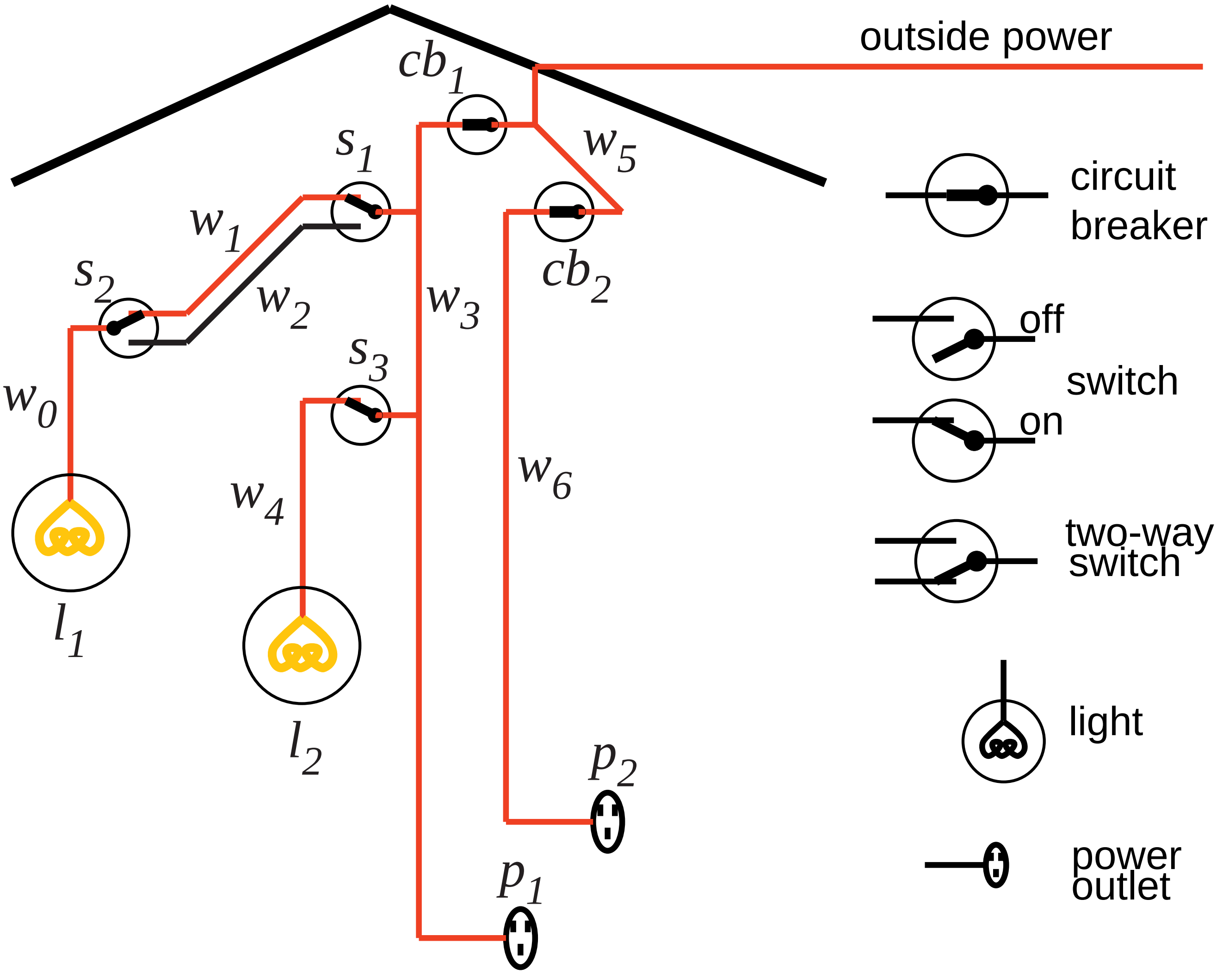
$$false \Leftarrow dark(L) \;\&\; lit(L).$$

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

- 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.

$$\begin{aligned}
&\mathit{false} \leftarrow \\
&\qquad \mathit{grade}(\mathit{St}, \mathit{Course}, \mathit{Gr}_1) \wedge \\
&\qquad \mathit{grade}(\mathit{St}, \mathit{Course}, \mathit{Gr}_2) \wedge \\
&\qquad \mathit{Gr}_1 \neq \mathit{Gr}_2.
\end{aligned}$$

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