Logic-based Truth Maintenance System (LTMS)

• Incrementally maintains consequences of a propositional theory $\Sigma$
  – incrementally manages addition and deletions from $\Sigma$
• $\Sigma$ is a set of propositional clauses
  – a clause is disjunction of propositional literals
  – a unit clause is a clause with exactly one disjunct
  – a literal is a proposition or the negation of a proposition

$$\neg \text{rain} \lor \neg \text{umbrella} \lor \text{dry}$$

– a clause can be read as an implication in different ways

$$\text{rain} \land \text{umbrella} \Rightarrow \text{dry}$$
$$\text{rain} \land \neg \text{dry} \Rightarrow \neg \text{umbrella}$$
Generic LTMS interface

• Updating the clauses in $\Sigma$
  – $\text{add-clause} \ (\text{clause}, \Sigma)$
  – $\text{delete-clause} \ (\text{clause}, \Sigma)$

• Propositional inference
  – $\text{consistent?} \ (\Sigma)$
  – $\text{follows-from?} \ (\text{literal}, \Sigma)$

• Justification structure
  – $\text{supporting-clause} \ (\text{literal}, \Sigma)$
  – $\text{supporting-literals} \ (\text{literal}, \Sigma)$
    • the $\text{supporting-clause}$ together with the $\text{supporting-literals}$ entail $\text{literal}$
    • each literal in $\text{supporting-literals}$ follows from $\Sigma$
    • $\bot$ is a special literal denoting a contradiction
Using the LTMS in diagnosis

- LTMS database $\Sigma$ contains clauses describing component behavior in each mode ($SD$)
- Search algorithm adds and deletes clauses corresponding to assumptions that a component is in a particular mode
  - checks that $\Sigma$ is consistent
  - justification structure is used to generate conflicts from an inconsistent $\Sigma$
**LTMS labels**

- The LTMS *labels* each proposition *true*, *false*, or *unknown*
  - if $p$ is labeled *true* (*false*), then $\Sigma$ logically entails $p$ ($\neg p$)
  - labeling algorithm is *sound*, but not necessarily *complete*

\[
\begin{array}{llll}
\Sigma & \neg p \lor \neg q \lor r & \neg s & s \lor p \\
& s \lor p & q & \neg u \lor v \\
& \neg u \lor v & u \lor v & \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Labels</th>
<th>p: true</th>
<th>u: unknown</th>
<th>q: true</th>
<th>v: unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>r: true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s: false</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conflicting clauses

- A *conflicting clause* is one in which all literals are labeled *false*
  
  \[ \neg p \lor \neg q \lor r \] is a conflicting clause if the labels are
  
  \[ p: true, q: true, r: false \]
- Existence of a conflicting clause means that \( \Sigma \) is *inconsistent*
- If \( \Sigma \) is inconsistent, \( \text{supporting-clause}(\bot, \Sigma) \) returns a conflicting clause and \( \text{supporting-literals}(\bot, \Sigma) \) returns the set of literals in that clause
Unit propagation at the fringe

- Unit propagation takes place at the fringe, which consists of all clauses that have
  - exactly one literal labeled unknown
  - all other literals labeled false
- Basic unit propagation algorithm
  - select a clause from the fringe and propagate until the fringe is empty or a conflicting clause is detected
Updating *fringe* and *conflicts*

- *fringe* and *conflicts* updated when a proposition’s label changes
  - only clauses in which the proposition occurs can update *fringe* or *conflicts*
- Membership in *fringe* and *conflicts* determined *incrementally*
  - track the count of literals in the clause labeled *unknown*
    - decrement (increment) the count when an *unknown* (true or false) literal becomes true or false (unknown)
  - track whether the clause is satisfied (i.e., contains a literal labeled true)

⇒ A clause is added to (removed from) the *fringe* if the *unknown* literal count becomes (changes from) 1 and it is not (or it is) satisfied

⇒ A clause is added to (removed from) the *conflicts* if the *unknown* literal count becomes (changes from) 0 and it is not (or it is) satisfied
LTMS after initialization

\[ C_7: \neg \text{ok} \lor \neg \text{uf} \]

\[ C_{10}: \text{ok} \]

\[ C_8: \neg \text{rf} \lor \neg \text{uf} \]

\[ C_{11}: \text{rf} \]

\[ C_1: \neg \text{nci} \lor \neg \text{a} \lor \text{nco} \]

\[ C_3: \neg \text{ok} \lor \text{a} \]

\[ C_6: \neg \text{ok} \lor \neg \text{rf} \]

\[ C_2: \neg \text{ia} \lor \text{nco} \]

\[ C_4: \neg \text{rf} \lor \text{ia} \]

\[ \text{Unknown literal count} \]
After propagation

\[ C_7: \neg ok \lor \neg uf \]

\[ C_{10}: ok \]

\[ C_8: \neg rf \lor \neg uf \]

\[ C_{11}: rf \]

\[ C_3: \neg ok \lor a \]

\[ C_6: \neg ok \lor \neg rf \]

\[ C_4: \neg rf \lor ia \]

\[ C_9: \neg a \lor \neg ia \]

\[ C_1: \neg nci \lor \neg a \lor nco \]

\[ C_2: \neg ia \lor nco \]

Unknown

\[ \text{nci} \]

\[ \text{uf false} \]

\[ \text{ok true} \]

\[ \text{false} \]

\[ \text{ia} \]

\[ \text{true} \]

\[ \text{unknown} \]

\[ \text{false} \]

\[ \text{true} \]

\[ \text{false} \]
Well-founded support

- Proposition supports generated by unit propagation form a directed acyclic graph
- Unit propagation produces well-founded support
- Non-well-founded support contains cycles in the support graph

\[ C_1: \neg x \lor y \]
\[ C_2: x \lor \neg y \]

Diagram:

- True
- \( x \)
- \( y \)
Implementing the generic interface

• \textit{consistent} \ ((\Sigma))
  – returns \textit{true} iff \Sigma has no conflicts after unit propagation terminates

• \textit{follows-from?} \ ((\text{literal}, \Sigma))
  – returns \textit{literal}’s label after unit propagation terminates

• \textit{supporting-clause} \ ((\text{literal}, \Sigma))
  \textit{supporting-literals} \ ((\text{literal}, \Sigma))
  – returns the clause and literals, respectively, that support \textit{literal} after unit propagation terminates
Incrementally modifying $\Sigma$

- **add-clause** ($clause, \Sigma$)
  - update $clause$’s unknown literal count and whether it is satisfied
  - update $\Sigma$’s fringe and conflicts appropriately
  - call $propagate\ (\Sigma)$

  $\Rightarrow$ need only do propagations (directly or indirectly) dependent on $clause$

- **delete-clause** ($clause, \Sigma$)
  - follow the support structure to set the label of all propositions (directly or indirectly) supported by $clause$ to unknown
    - update $\Sigma$’s fringe and conflicts as labels are changed
  
  $\Rightarrow$ only propagations (directly or indirectly) dependent on $clause$ are undone
  - call $propagate\ (\Sigma)$
Before deleting $C_{11}$

- $C_7$: $\neg ok \lor \neg uf$
- $C_{10}$: $ok$
- $C_8$: $\neg rf \lor \neg uf$
- $C_{11}$: $rf$
- $C_2$: $\neg ia \lor nco$
- $C_1$: $\neg nci \lor \neg a \lor nco$
- $C_9$: $\neg a \lor \neg ia$
- $C_4$: $\neg rf \lor ia$
- $C_6$: $\neg ok \lor \neg rf$
- $C_3$: $\neg ok \lor a$
- $C_5$: unknown

The diagram shows the relationships and implications between the variables and clauses before deleting $C_{11}$. The truth values for each variable and clause are indicated, with red boxes highlighting the clauses that become true after deletion.