**Sugar 2.0 in HOL**

a deep embedding

Mike Gordon

(with help from Cindy Eisner and Dana Fisman of IBM)

IBM’s Sugar 2.0 selected by Accellera as the industry standard property language

- LTL based, but incorporates features from ITL (regular expressions) and CTL
  
  \[ \{r_1\} \rightarrow \{r_2\}, \{r_1 \& \& r_2\} \text{ (ITL)}, \quad X f, [f_1 U f_2] \text{ (LTL)}, \quad EX f, EIF [f_1 U f_2] \text{ (CTL)} \]
- supports infinite paths (for model checking) and finite paths (for simulation run checking)
- has both clocked and unclocked semantics (equivalent for trivial clocks)
  
  \[ \pi \models f \quad \text{if formula } f \text{ holds for (finite or infinite) path } \pi \text{ when weakly clocked on } c \]
  
  \[ \pi \models f \quad \text{if formula } f \text{ holds for (finite or infinite) path } \pi \text{ when strongly clocked on } c \]
- most constructs are defined from a small kernel (name ‘sugar’ from ‘syntactic sugar’)
  
  \[ b[i] = \begin{cases} b[i]; & \text{if } b[i] \neq b[i]; s[i]; \text{if } b[i]; \end{cases} \]
  
  \[ \{r_1\} \rightarrow \{r_2\} \text{, } \{r_1 \& \& b[i] = 0\}; b \]
  
  \[ [f_1 W f_2] = [f_1 U f_2] \lor GF, \text{ next_event}(b)(f) = \neg b W b \land f \]

**Semantics in HOL’s classical higher order logic is a straightforward deep embedding**

<table>
<thead>
<tr>
<th>Sugar syntax</th>
<th>HOL representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ f_1 U f_2 ]</td>
<td>[ \text{F_UNTIL}(f_1, f_2) ]</td>
</tr>
</tbody>
</table>

**Typical examples of minor errors found by attempting to prove ‘sanity checking’ properties**

<table>
<thead>
<tr>
<th>Original semantics</th>
<th>Corrected semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \pi \models b \quad \text{if there exists } i \text{ s.t. } \pi^i \models \neg c \text{ and for every } j \text{ s.t. } i \leq j &lt; k \text{ and } \pi^j \models c; \pi^j \models f_2 ]</td>
<td>[ \pi \models b \quad \text{for every } i \text{ s.t. } \pi^i \models \neg c \text{ and for every } j \text{ s.t. } \pi^j \models c, ]</td>
</tr>
<tr>
<td>[ \pi \models {r_1} \rightarrow {r_2} \quad \text{either for every } j \text{ such that } \pi^j \models r_1 \text{ there exists } k \text{ such that } \pi^{j+k} \models r_2 ]</td>
<td>[ \pi \models {r_1} \rightarrow {r_2} \quad \text{either for every } j \text{ such that } \pi^j \models r_1 \text{ or for every } k \text{ there exists a finite word } w \text{ such that } \pi^{j+k}w \models r_2 ]</td>
</tr>
</tbody>
</table>

**HOL deduction can be used to derive and verify proof rules**

- McMillan’s ‘circular inference rule’
  
  ‘\[ [f_2 U f_1] \text{ means } f_2 \text{ up to } t-1 \text{ implies } f_1 \text{ at } t, \]’
  
  ‘\[ [f_1 U f_2] \text{ means } f_1 \text{ up to } t-1 \text{ implies } f_2 \text{ at } t, \]’
  
  so:
  
  \[ \neg[f_2 U f_1], \quad \neg[f_1 U f_2] \quad G(f_1 \land f_2) \]

- An iteration rule (c.f. Hoare Logic while-rule)
  
  assume functions \( f, b \) and \( g \) satisfy:
  
  \[ \forall x. f(x) = \text{ if } b(x) \text{ then } f(g(x)) \text{ else } x \] and ‘\( \langle x \rangle \)’ means ‘the current state is \( x \)’, then:
  
  \[ \forall x. G\langle x \rangle \rightarrow X!g(x)\rangle \]
  
  \[ \forall x. \langle x \rangle \rightarrow \text{next_event}(\neg b)\langle f(x)\rangle \]