Using HOL to study *Sugar 2.0* semantics

Michael J. C. Gordon

University of Cambridge Computer Laboratory
William Gates Building, JJ Thomson Avenue, Cambridge CB3 0FD, U.K.
Email: `mjcg@cl.cam.ac.uk` Web: `http://www.cl.cam.ac.uk/~mjcg`

July 5, 2002

Abstract. The Accellera standards-promoting organisation has selected *Sugar 2.0*, IBM's formal specification language, as a standard that it says will drive assertion-based verification. *Sugar 2.0* combines aspects of Interval Temporal Logic (ITL), Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) into a property language suitable for both formal verification and use with simulation test benchs. As industrial strength languages go it is remarkably elegant, consisting of a small kernel conservatively extended by numerous definitions.
We are constructing a semantic embedding of *Sugar 2.0* in the version of higher order logic supported by the HOL system. To 'sanity check' the semantics we tried to prove some simple properties and as a result a few bugs were discovered. Further analysis may well reveal more.
We are contemplating a variety of applications of the mechanised semantics, including the exploitation of existing work to build a Sugar model checker inside HOL. In the longer term we want to investigate the use of theorem proving to reason about models with infinite state spaces, which might involve developing extensions of *Sugar 2.0*.

1 Background on Accellera and Sugar

The Accellera organisation's website has their mission statement:

> *To improve designers' productivity, the electronic design industry needs a methodology based on both worldwide standards and open interfaces. Accellera was formed in 2000 through the unification of Open Verilog International and VHDL International to focus on identifying new standards, development of standards and formats, and to foster the adoption of new methodologies.*
> *Accellera's mission is to drive worldwide development and use of standards required by systems, semiconductor and design tools companies, which enhance a language-based design automation process. Its Board of Directors guides all the operations and activities of the organisation and is comprised of representatives from ASIC manufacturers, systems companies and design tool vendors.*

Faced with a plethora of syntactically and semantically incompatible formal property languages, Accellera initiated a process of selecting a standard property language to "drive assertion-based verification".

Four contributions were initially considered

- Motorola's CBV language;
- IBM's Sugar (the language of its RuleBase FV toolset);
- Intel's ForSpec;
- Verisity's *e* language (the language of the Specman Elite testbench).

After a combination of discussion and voting, some details of which can be viewed on the web[1], attention was narrowed down to Sugar and CBV, and then in April 2002 a vote[2] selected IBM's submission, *Sugar 2.0*.

Sugar 2.0 is primarily an LTL-based language that is a successor to the CTL-based *Sugar 1* [1]. A key idea of both languages is the use of ITL-like [3] constructs called *Sugar Extended Regular Expressions*. *Sugar 2.0* retains CTL constructs in its *Optional Branching Extension* (OBE), but this is de-emphasised in the defining document.

Besides moving from CTL to LTL, *Sugar 2.0* supports clocking and finite paths. Clocking allows one to specify on which clock edges signals are sampled at (i.e. it defines the 'next event' for each signal). The finite path semantics allows properties to be interpreted on simulation runs, as in test-bench tools like Specman and Vera.

The addition of clocking and finite path semantics makes the *Sugar 2.0* semantics more than twice as complicated as the *Sugar 1* semantics. However, for a real 'industry standard' language *Sugar 2.0* is still remarkably simple, and it was routine to define the abstract syntax and semantics of the whole language in HOL [2].

In the rest of this paper we start by discussing the point of embedding Sugar in HOL. Next we briefly review semantic embedding, illustrating the ideas on simplified semantics of fragments of *Sugar 2.0*. We then give the complete semantics of *Sugar 2.0*, and finally we discuss our small achievements so far in analysing the semantics using the HOL system, including a discussion of the bugs found.

2 Why embed Sugar in HOL?

There are several justifications for the work described here. This project has only just started and its goals are still being defined. Current motivations include the following.

2.1 Sanity checking and proving meta-theorems

By formalising the semantics and passing it through a parser and type-checker one achieves a first level of sanity checking of the definition. One also exposes possible ambiguities, fuzzy corner cases etc (e.g. see Section 4.2). The process is also very educational for the formaliser and a good learning exercise.

There are a number of meta-theorems one might expect to be true, and proving them with a theorem prover provides a further and deeper kind of sanity checking. In the case of *Sugar 2.0*, such meta-theorems include showing that expected

[1] http://www.eda.org/vfv/hm/

[2] http://www.eda.org/vfv/hm/0795.html

simplifications to the semantics occur if there is no non-trivial clocking, that different semantics of clocking are equivalent and that if finite paths are ignored then the standard 'text-book semantics' results. Such meta-theorems are generally mathematically shallow, but full of tedious details – i.e. ideal for automated theorem proving. See Section 5 for what we have proved so far. It's not much, but we have already found minor bugs in the semantics!

A key feature of the Sugar approach is to have a small kernel and a large number of definitions. Using a theorem prover, the definitions can be validated by proving that they achieve the correct semantics. See, for example, the analysis of `FirstRise` and `NextRise` in Section 5 (these are not official Sugar definitions, but the analysis here illustrates the idea of validating definitions).

2.2 Develop a machine readable semantics

The current *Sugar 2.0* document is admirably clear, but it is informal mathematics presented as typeset text. Tool developers could benefit from a machine readable version. One might think of using some standard representation of mathematical content, like MathML[3], however there is currently not much mathematically sophisticated tool support for such XML-based representations.

Higher order logic is a widely used formalisation medium (versions of higher order logic are used by HOL, Isabelle/HOL, PVS, NuPrl and Coq) and the semantic embedding of model-checkable logics in HOL is standard [5, 4]. Once one has a representation in it then representations in other formats should be straightforward to derive.

2.3 Research using our local tools

We are contemplating developing semantically-based reasoning and checking infrastructure in HOL to support *Sugar 2.0*, and a prerequisite for this is to have a 'golden semantics' to which application specific semantics can be proved equivalent.

One area of research that we have an interest in is the development of property languages that support data operations and can have variables ranging over infinite data-types like numbers (e.g. including reals and complex numbers for DSP applications). Some sort of mixture of Hoare Logic and *Sugar 2.0* is being contemplated (rather vaguely, it must be admitted). Developing the language by extending an existing semantics is a way to ensure some 'backward compatibility'. Also, we might wish to prove sanity checking meta-theorem about our extended language, e.g. that it collapses to *Sugar 2.0* when there are no infinite types.

2.4 Education

Both semantic embedding and property specification are taught as part of the Computer Science undergraduate course at Cambridge[4], and being able to illustrate the ideas on a real example like *Sugar 2.0* is pedagogically valuable.

[3] http://www.w3.org/Math/

[4] http://www.cl.cam.ac.uk/users/mjcg/Teaching/SpecVer2/SpecVer2.html

The semantic embadding of *Sugar 2.0* in the HOL system is an interesting case study. It nicely illustrates some issues in making total functional definitions, and the formal challenges attempted so far provide insight into how to perform structural induction using the built-in tools. Thus *Sugar 2.0* has educational potential for training HOL users. In fact, the semantics described in this paper is an example distributed with HOL.[5]

3 Review of semantic embedding in higher order logic

Higher order logic is an extension of first-order predicate calculus that allows quantification over functions and relations. It is a natural language for formalising informal set theoretic specifications (indeed, it is usually more natural than formal first-order set theories, like ZF). We hope that the HOL notation that follows is sufficiently close to standard informal notation that it needs no systematic explanation.

We use Church's λ-notation for denoting functions: a 'lambda-term' like $\lambda x.\ t$, where x is a variable and t a term, denotes the function that maps a value v to the result of substituting v for the variable x in t (the infix notation $x \mapsto t$ is sometimes used instead of $\lambda x.\ t$). If P is a function that returns a truth-value (i.e. a predicate), then P can be thought of a set, and we write $x \in P$ to mean $P(x)$ is true. Note that $\lambda x.\ \cdots x \cdots$ corresponds to the set abstraction $\{x \mid\ \cdots x \cdots\}$. We write $\forall x \in P.\ Q(x)$, $\exists x \in P.\ Q(x)$ to mean $\forall x.\ P(x) \Rightarrow Q(x)$, $\exists x.\ P(x) \wedge Q(x)$, respectively.

To embed[6] a language in HOL one first defines constructors for all the syntactic constructs of the language. This is the 'abstract syntax' and provides a representation of parse trees as terms in the logic. The semantics is then given by defining a semantic function that recursively maps each construct to a representation of its meaning.

For *Sugar 2.0*, a model $\mathtt{M}$ is a quintuple $(\mathtt{S}_\mathtt{M}, \mathtt{S0}_\mathtt{M}, \mathtt{R}_\mathtt{M}, \mathtt{P}_\mathtt{M}, \mathtt{L}_\mathtt{M})$, where $\mathtt{S}_\mathtt{M}$ is a set of states, $\mathtt{S0}_\mathtt{M}$ is the subset of initial states, $\mathtt{R}_\mathtt{M}$ is a transition relation (so $\mathtt{R}_\mathtt{M}(s, s')$ means s' is a possible successor state to s), $\mathtt{P}_\mathtt{M}$ is a set of atomic propositions, and $\mathtt{L}_\mathtt{M}$ is a valuation that maps a state to the set of atomic propositions that hold at the state (so $\mathtt{L}\ s\ \mathtt{p}$ is true iff atomic proposition $\mathtt{p}$ is true in state s).

The syntax of boolean expressions $\mathtt{b}$ built from atomic propositions (ranged over by $\mathtt{p}$) using negation ($\neg$) and conjunction ($\wedge$) is given by:

$$\begin{array}{lll} \mathtt{b} ::= & \mathtt{p} & \text{(Atomic formula)} \\ & \mid \neg\mathtt{b} & \text{(Negation)} \\ & \mid \mathtt{b}_1\ \wedge\ \mathtt{b}_2 & \text{(Conjunction)} \end{array}$$

in HOL this is defined by a recursive type definition of a syntactic type of boolean expressions.

[5] http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/hol/hol98/examples/Sugar2/

[6] We shall only be concerned with so called 'deep embeddings' here.

Let l range over predicates on P_M, called "truth assignments" in the Sugar documentation. The semantics of boolean expressions is given by defining a semantic function B_SEM such that B_SEM M l b if true iff b is built from propositions in P_M and it is true with respect to the truth assignment l.
If we write (M, l $\models$ b) for B_SEM M l b then the semantics is given by

$$
\begin{array}{l}
((\mathtt{M},\ \mathtt{l} \models \mathtt{p}) \quad = \mathtt{p} \in \mathtt{P_M} \wedge \mathtt{p} \in \mathtt{l})\\
\wedge\\
((\mathtt{M},\ \mathtt{l} \models \mathtt{T}) \quad = \mathtt{T})\\
\wedge\\
((\mathtt{M},\ \mathtt{l} \models \neg\mathtt{b}) \quad = \neg(\mathtt{M},\ \mathtt{l} \models \mathtt{b}))\\
\wedge\\
((\mathtt{M},\ \mathtt{l} \models \mathtt{b1} \wedge \mathtt{b2}) = (\mathtt{M},\ \mathtt{l} \models \mathtt{b1}) \wedge (\mathtt{M},\ \mathtt{l} \models \mathtt{b2}))
\end{array}
$$

Before looking at the full official semantics of *Sugar 2.0*, we first consider a simplified semantics in which there is no clocking, and paths are always infinite. We consider separately the parts of *Sugar 2.0* corresponding to Interval Temporal Logic (ITL), Linear Temporal Logic (LTL) and Computation Tree Logic (CTL).

3.1 ITL: Sugar Extended Regular Expressions (SEREs)

Interval Temporal Logic (ITL) provides formulas that are true or false of intervals of states. Here we just consider finite intervals, though recent formulations of ITL[7] allow intervals to be infinite. For Sugar we only need to consider ITL formulas, as there are no constructs corresponding to ITL expressions (expressions map intervals to values). Providing more elaborate ITL constructs in Sugar strikes us as an interesting research topic.
The Sugar subset corresponding to ITL is called *Sugar Extended Regular Expressions* (SEREs). If r ranges over SEREs and p ranges over a set P_M of atomic propositions, then the syntax is given by:

$$
\begin{array}{ll}
\mathtt{r} ::= \mathtt{p} & \text{(Atomic formula)}\\
\quad \mid \{\mathtt{r}_1\}\ \mathtt{|}\ \{\mathtt{r}_2\} & \text{(Disjunction)}\\
\quad \mid \mathtt{r}_1\ ;\ \mathtt{r}_2 & \text{(Concatenation)}\\
\quad \mid \mathtt{r}_1\ :\ \mathtt{r}_2 & \text{(Fusion: ITL's chop)}\\
\quad \mid \{\mathtt{r}_1\}\ \&\&\ \{\mathtt{r}_2\} & \text{(Length matching conjunction)}\\
\quad \mid \{\mathtt{r}_1\}\ \&\ \{\mathtt{r}_2\} & \text{(Flexible matching conjunction)}\\
\quad \mid \mathtt{r}[*] & \text{(Repeat)}
\end{array}
$$

The semantics of SEREs is given by defining a semantic function S_SEM such that S_SEM M w r if true iff w is in the language of the extended regular expression r. We write (M, w $\models$ r) for S_SEM M w r.
If wlist is a list of lists then Concat wlist is the concatenation of the lists in wlist and if P is some predicate then Every P wlist means that P(w) holds for every w in wlist.

[7] http://www.cms.dmu.ac.uk/~cau/itlhomepage/

```
((M, w ⊨ b) =
  ∃l. (w = [l]) ∧ (M, l ⊨ b))
∧
((M, w ⊨ r1;r2) =
  ∃w1 w2. (w = w1 w2) ∧ (M, w1 ⊨ r1) ∧ (M, w2 ⊨ r2))
∧
((M, w ⊨ r1:r2) =
  ∃w1 w2 l. (w = w1 [l] w2) ∧
            (M, (w1 [l]) ⊨ r1) ∧ (M, ([l] w2) ⊨ r2))
∧
((M, w ⊨ {r1}|{r2}) =
  (M, w ⊨ r1) ∨ (M, w ⊨ r2))
∧
((M, w ⊨ {r1}&&{r2}) =
  (M, w ⊨ r1) ∧ (M, w ⊨ r2))
∧
((M, w ⊨ {r1}&{r2}) =
  ∃w1 w2. (w = w1 w2) ∧
          (((M, w ⊨ r1) ∧ (M, w1 ⊨ r2))
           ∨
           ((M, w ⊨ r2) ∧ (M, w1 ⊨ r1))))
∧
((M, w ⊨ r[*]) =
  ∃wlist. (w = Concat wlist) ∧ Every (λw. (M, w ⊨ r)) wlist)
```

3.2 LTL: Sugar Foundation Language (FL)

Sugar 2.0 has a kernel combining standard LTL notation with a less standard `abort` operation and some constructs using SEREs. The suffix "`!`" found on some constructs indicates that these are 'strong' (i.e. liveness-enforcing) operators. The distinction between strong and weak operators is given in the semantics of full *Sugar 2.0* in Section 4. Numerous additional notations are introduced by definitions (which are conservative extensions of the language, and can be formalised as definitions in HOL).

$\mathtt{f} ::= \mathtt{p}$	(Atomic formula)
$\mid \neg\mathtt{f}$	(Negation)
$\mid \mathtt{f}_1 \wedge \mathtt{f}_2$	(Conjunction)
$\mid \mathtt{X!f}$	(Successor)
$\mid [\mathtt{f}_1\ \mathtt{U}\ \mathtt{f}_2]$	(Until)
$\mid \{\mathtt{r}\}(\mathtt{f})$	(Suffix implication)
$\mid \{\mathtt{r}_1\}\ \mathtt{\vert\text{-}>}\ \{\mathtt{r}_2\}\mathtt{!}$	(Strong suffix implication)
$\mid \{\mathtt{r}_1\}\ \mathtt{\vert\text{-}>}\ \{\mathtt{r}_2\}$	(Weak suffix implication)
$\mid \mathtt{f\ abort\ b}$	(Abort)

Being LTL, the semantics of FL formulas is defined with respect to a path π, which (in the simplified semantics here) is a function from the natural numbers to states.

We define a semantic function F_SEM such that F_SEM M π f means FL formula f is true of path π. We write (M, π $\models$ r) for F_SEM M π f.
Note that in the semantics below it is not assumed that paths π are necessarily computations (i.e. satisfy Path M π, as defined in Section 4.6). This is important for the abort construct (where the $\exists\pi'$ quantifies over all paths).
The notation π_i denotes the i-th state in the path (i.e. $\pi(i)$); π^i denotes the 'i-th tail' of π – the path obtained by chopping i elements off the front of π (i.e. $\pi^i = \lambda n.\ \pi(n{+}i)$). $\pi^{(i,j)}$ denotes the finite sequence of states from i to j in π, i.e. $\pi_i\pi_{i+1}\cdots\pi_j$. The juxtaposition $\pi^{(i,j)}\pi'$ denotes the path obtained by concatenating the finite sequence $\pi^{(i,j)}$ on to the front of the path π'.
The function $\hat{\mathtt{L}}_{\mathtt{M}}$ denotes the point-wise extension of $\mathtt{L}_{\mathtt{M}}$ to finite sequences of states (i.e. MAP $\mathtt{L}_{\mathtt{M}}$ in HOL and functional programming notation).

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \models\ \mathtt{b}) = (\mathtt{M},\ \mathtt{L}_{\mathtt{M}}(\pi_0)\ \models\ \mathtt{b}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \neg\mathtt{f}) = \neg(\mathtt{M},\ \pi\ \models\ \mathtt{f}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \mathtt{f1} \wedge \mathtt{f2}) = (\mathtt{M},\ \pi\ \models\ \mathtt{f1}) \wedge (\mathtt{M},\ \pi\ \models\ \mathtt{f2}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \mathtt{X!\,f}) = (\mathtt{M},\ \pi^{1}\ \models\ \mathtt{f}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ [\mathtt{f1\ U\ f2}]) =\\
\quad \exists\mathtt{k}.\ (\mathtt{M},\ \pi^{\mathtt{k}}\ \models\ \mathtt{f2}) \wedge \forall\mathtt{j}.\ \mathtt{j} < \mathtt{k} \Rightarrow (\mathtt{M},\ \pi^{\mathtt{j}}\ \models\ \mathtt{f1}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \{\mathtt{r}\}(\mathtt{f})) =\\
\quad \forall\mathtt{j}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(0,\mathtt{j})}))\ \models\ \mathtt{r}) \Rightarrow (\mathtt{M},\ \pi^{\mathtt{j}}\ \models\ \mathtt{f}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \{\mathtt{r1}\}\mathtt{|\text{-}>}\{\mathtt{r2}\}!) =\\
\quad \forall\mathtt{j}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(0,\mathtt{j})}))\ \models\ \mathtt{r1})\\
\qquad\quad \Rightarrow \exists\mathtt{k}.\ \mathtt{j} \leq \mathtt{k} \wedge (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\ \models\ \mathtt{r2}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \{\mathtt{r1}\}\mathtt{|\text{-}>}\{\mathtt{r2}\}) =\\
\quad \forall\mathtt{j}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(0,\mathtt{j})}))\ \models\ \mathtt{r1})\\
\qquad\quad \Rightarrow (\exists\mathtt{k}.\ \mathtt{j} \leq \mathtt{k} \wedge (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\ \models\ \mathtt{r2}))\\
\qquad\qquad \vee\\
\qquad\qquad \forall\mathtt{k}.\ \mathtt{j} \leq \mathtt{k} \Rightarrow \exists\mathtt{w}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\mathtt{w}\ \models\ \mathtt{r2}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \mathtt{f\ abort\ b}) =\\
\quad ((\mathtt{M},\ \pi\ \models\ \mathtt{f}\ )\\
\quad \vee\\
\quad \exists\mathtt{j}\ \pi'.\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models\ \mathtt{b}) \wedge (\mathtt{M},\ \pi^{(0,\mathtt{j}-1)}\pi'\ \models\ \mathtt{f})))
\end{array}
$$

3.3 CTL: Sugar Optional Branching Extension (OBE)

The syntax of the *Sugar 2.0* OBE is completely standard. The syntax of the OBE formulas is:

$$
\begin{array}{lll}
\texttt{f} ::= & \texttt{p} & \text{(Atom)} \\
 & \mid \neg\texttt{f} & \text{(Negation)} \\
 & \mid \texttt{f}_1 \wedge \texttt{f}_2 & \text{(Conjunction)} \\
 & \mid \mathbf{EX}\texttt{f} & \text{(Some successors)} \\
 & \mid \mathbf{E}[\texttt{f}_1\ \mathbf{U}\ \texttt{f}_2] & \text{(Until – along some path)} \\
 & \mid \mathbf{EG}\texttt{f} & \text{(Always on some path)}
\end{array}
$$

For the semantics, define `Path M` π to be true iff π is a computation of `M`:

$$\texttt{Path M}\ \pi\ \texttt{=}\ \forall\texttt{n.}\ \texttt{R}_M(\pi_n,\ \pi_{n+1})$$

The semantic function `O_SEM` is defined so that `O_SEM M s f` is true iff `f` is true of `M` at state `s`. Write (`M, s` $\models$ `f`) for `O_SEM M s f`, and then the semantics of the OBE is defined by:

$$
\begin{array}{l}
((\texttt{M, s} \models \texttt{b}) = (\texttt{M, L}_\texttt{M}(\texttt{s}) \models \texttt{b})) \\
\wedge \\
((\texttt{M, s} \models \neg\texttt{f}) = \neg((\texttt{M, s} \models \texttt{f}))) \\
\wedge \\
((\texttt{M, s} \models \texttt{f1} \wedge \texttt{f2}) = (\texttt{M, s} \models \texttt{f1}) \wedge (\texttt{M, s} \models \texttt{f2})) \\
\wedge \\
((\texttt{M, s} \models \texttt{EX f}) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge (\pi_0 = \texttt{s}) \wedge (\texttt{M},\ \pi_1 \models \texttt{f})) \\
\wedge \\
((\texttt{M, s} \models [\texttt{f1 U f2}]) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge (\pi_0 = \texttt{s}) \wedge \\
\qquad (\texttt{M},\ \pi_\texttt{k} \models \texttt{f2}) \wedge \forall\texttt{j. j} < \texttt{k} \Rightarrow (\texttt{M},\ \pi_\texttt{j} \models \texttt{f1})) \\
\wedge \\
((\texttt{M, s} \models \texttt{EG f}) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge (\pi_0 = \texttt{s}) \wedge \forall\texttt{j.}\ (\texttt{M},\ \pi_\texttt{j} \models \texttt{f}))
\end{array}
$$

4 Full *Sugar 2.0* semantics in higher order logic

The semantics that follows is derived from of a deep semantic embedding of *Sugar 2.0* in higher order logic (HOL-4 version). The official *Sugar 2.0* semantics can be found in the Accellera submission document [6]

http://www.haifa.il.ibm.com/projects/verification/sugar/literature.html

Corresponding to Appendix A.1 of the Sugar documentation we have defined types `bexp`, `sere`, `fl` and `obe` in the HOL logic to represent the syntax of Boolean Expressions, Sugar Extended Regular Expressions (SEREs), formulas

of the Sugar Foundation Language (FL) and formulas of the Optional Branching Extension (OBE), respectively.
Corresponding to Appendix A.2 of the Sugar documentation we have defined semantic functions `B_SEM`, `S_SEM`, `F_SEM` and `O_SEM` that interpret boolean expressions, SEREs, FL formulas and OBE formulas, respectively.
The HOL definitions can be seen in

http://www.cl.cam.ac.uk/~mjcg/Sugar

In the next two sub-sections we discuss clocking and finite paths. In the remaining four sub-sections we give manually typeset versions of the HOL definitions of the semantic functions. The typesetting was done as algorithmically as I could by editing the HOL sources. I hope to have avoided transcription errors, but this cannot be guaranteed. Following some proof activity, changes to the original HOL semantics have been made, as discussed in Section 5.

4.1 Clocking

If `b` is a boolean expression, then the SERE `b@clk` recognises a sequence of states in which `b` is true on the next rising edge of `clk`. Thus `b@clk` behaves like {`¬clk[*]; clk ∧ b`}.
One can also clock formulas (`f@clk`), and there may be several clocks. Consider: [8]

```
G(req_in -> X(req_out@clkb))@clka
```

this means that the entire formula is clocked on clock `clka`, except that signal `req_out` is clocked on `clkb`. Clocks do not "accumulate", so the signal `req_out` is only clocked by `clkb`, not by both clocks. Thus `clkb` "protects" `req_out` from the main clock, `clka`, i.e.:

```
req_out@clkb@clka = req_out@clkb
```

This meaning of clocking prevents us simply defining:

```
req_out@clkb = [¬clkb U (clkb ∧ req_out)]
```

since if this were the definition of `req_out@clkb` then we would be forced to have:

```
req_out@clkb@clka = [¬clkb U (clkb ∧ req_out)]@clka
```

when we actually want

```
req_out@clkb@clka = req_out@clkb
```

Thus we cannot just rewrite away clocking annotations using equational reasoning. However, if one starts at the outside and works inwards, then one can systematically compile away clocking. The rules for doing this are given in the *Sugar 2.0* documentation as part of the implementation of formal verification.

[8] The discussion of clocking here is based on email communication with Cindy Eisner.

The official semantics uses a different approach in which the currently active clock is an argument of the semantic function used to interpret SEREs and formulas. Proving this approach equivalent to compiling away clocks, followed by a simpler unclocked semantics, is one of the formal challenges to which we hope to submit the semantics.

4.2 Finite paths

Sugar 2.0 gives a semantics to formulas for both finite and infinite paths. To represent this, we model a path as being either a non-empty[9] finite list of states or a function from natural numbers to states and define a predicate `finite` to test if a path is a finite list. The function `length` gives the length of a finite path (it is not defined on paths for which `finite` is not true).

We interpret the official semantics locution

"for every $j <$ `length`(π): $\cdots\ j\ \cdots$"

as meaning

"for every j: (`finite` π implies $j <$ `length` π) implies $\cdots\ j\ \cdots$"

and we interpret the official semantics locution

"there exists $j <$ `length`(π) s.t. $\cdots\ j\ \cdots$"

as meaning

"there exists j s.t. (`finite` π implies $j <$ `length` π) and $\cdots\ j\ \cdots$"

Define `pl` $\pi\ n$ to mean that if π is finite then n is less than the length of π, i.e. the predicate `pl` is defined by

$\mathtt{pl}\ \pi\ n\ =\ \mathtt{finite}\ \pi\ \Rightarrow\ n < \mathtt{length}\ \pi$

We can then write "$\forall i \in \mathtt{pl}\ \pi.\ \cdots\ i\ \cdots$" and "$\exists i \in \mathtt{pl}\ \pi.\ \cdots\ i\ \cdots$" for the locutions above. The name "`pl`" is short for "path length"

Here is a version of the unclocked FL semantics that allows paths to be finite.

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \models\ \mathtt{b}) = (\mathtt{M},\ \mathtt{L_M}(\pi_0)\ \models\ \mathtt{b}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \neg\mathtt{f}) = \neg(\mathtt{M},\ \pi\ \models\ \mathtt{f})))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \mathtt{f1} \wedge \mathtt{f2}) = (\mathtt{M},\ \pi\ \models\ \mathtt{f1})\ \wedge\ (\mathtt{M},\ \pi\ \models\ \mathtt{f2}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \mathtt{X!\,f}) = \mathtt{pl}\ \pi\ 1\ \wedge\ (\mathtt{M},\ \pi^1\ \models\ \mathtt{f}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ [\mathtt{f1\ U\ f2}]) =\\
\quad \exists \mathtt{k} \in \mathtt{pl}\,\pi.\\
\quad\ (\mathtt{M},\ \pi^{\mathtt{k}}\ \models\ \mathtt{f2})\ \wedge\ \forall \mathtt{j} \in \mathtt{pl}\,\pi.\ \mathtt{j}\ <\ \mathtt{k}\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models\ \mathtt{f1}))\\
\wedge\\
((\mathtt{M},\ \pi\ \models\ \{\mathtt{r}\}(\mathtt{f})) =\\
\quad \forall \mathtt{j} \in \mathtt{pl}\,\pi.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(0,\mathtt{j})}))\ \models\ \mathtt{r})\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models\ \mathtt{f}))
\end{array}
$$

[9] The need for finite paths to be non-empty arose when trying to prove some properties. This requirement does not seem to be explicit in the Accellera specification.

$\wedge$
((M, π $\models$ {r1}|->{r2}!) =
  $\forall$j $\in$ pl π. (M, ($\hat{\text{L}}_{\text{M}}$ ($\pi^{(0,j)}$)) $\models$ r1)
    $\Rightarrow$ $\exists$k $\in$ pl π. j $\leq$ k $\wedge$ (M, ($\hat{\text{L}}_{\text{M}}$ ($\pi^{(j,k)}$)) $\models$ r2))
$\wedge$
((M, π $\models$ {r1}|->{r2}) =
  $\forall$j $\in$ pl π. (M, ($\hat{\text{L}}_{\text{M}}$ ($\pi^{(0,j)}$)) $\models$ r1)
    $\Rightarrow$ ($\exists$k $\in$ pl π. j $\leq$ k $\wedge$ (M, ($\hat{\text{L}}_{\text{M}}$ ($\pi^{(j,k)}$)) $\models$ r2))
      $\vee$
      $\forall$k $\in$ pl π. j $\leq$ k $\Rightarrow$ $\exists$w. (M, ($\hat{\text{L}}_{\text{M}}$ ($\pi^{(j,k)}$))w $\models$ r2))
$\wedge$
((M, π $\models$ f abort b) =
  ((M, π $\models$ f)
   $\vee$
   $\exists$j $\in$ pl π.
    0 < j $\wedge$ $\exists\pi$'. (M, π^{j} $\models$ b) $\wedge$ (M, $\pi^{(0,j-1)}\pi'$ $\models$ f)))

This semantics was initially derived from an existing unpublished semantics of unclocked FL formulas[10].
The following four sub-sections are the manually typeset HOL semantics of *Sugar 2.0*.

4.3 Boolean expressions

The semantics below is identical to that given earlier in Section 3.

((M, l $\models$ p)   = $p \in \text{P}_M \wedge p \in$ l)
$\wedge$
((M, l $\models$ T)   = T)
$\wedge$
((M, l $\models$ $\neg$b)   = $\neg$(M, l $\models$ b))
$\wedge$
((M, l $\models$ b1 $\wedge$ b2) = (M, l $\models$ b1) $\wedge$ (M, l $\models$ b2))

4.4 Sugar Extended Regular Expressions

The semantics of SEREs expressions is given by defining a semantic function `S_SEM` such that `S_SEM M w c r` if true iff `w` is in the language of the extended regular expression `r` clocked with `c`.
We write (M, w $\overset{\text{c}}{\models}$ r) for `S_SEM M w c r`.
If `wlist` is a list of lists then `Concat wlist` is the concatenation of the lists in `wlist` and if P is some predicate then `Every` P `wlist` means that P(w) holds for every w in `wlist`.

[10] Personal communication from Cindy Eisner.

$$
\begin{array}{l}
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{b}) = \\
\quad \exists \mathtt{n}.\ \mathtt{n} \geq 1 \hfill \wedge \\
\qquad (\mathrm{length}\ \mathtt{w} = \mathtt{n}) \hfill \wedge \\
\qquad (\forall \mathtt{i}.\ 1 \leq \mathtt{i} \wedge \mathtt{i} < \mathtt{n} \Rightarrow (\mathtt{M},\ \mathtt{w}_{\mathtt{i}-1} \models \neg \mathtt{c}) \wedge \\
\qquad (\mathtt{M},\ \mathtt{w}_{\mathtt{n}-1} \models \mathtt{c} \wedge \mathtt{b})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1;r2}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}.\ (\mathtt{w} = \mathtt{w1}\,\mathtt{w2}) \wedge (\mathtt{M},\ \mathtt{w1}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w2}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1:r2}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}\ \mathtt{l}.\ (\mathtt{w} = \mathtt{w1}\,[\mathtt{l}]\,\mathtt{w2}) \wedge \\
\qquad\qquad (\mathtt{M},\ (\mathtt{w1}\,[\mathtt{l}])\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1}) \wedge (\mathtt{M},\ ([\mathtt{l}]\,\mathtt{w2})\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \{\mathtt{r1}\}|\{\mathtt{r2}\}) = \\
\quad (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1}) \vee (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \{\mathtt{r1}\}\&\&\{\mathtt{r2}\}) = \\
\quad (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \{\mathtt{r1}\}\&\{\mathtt{r2}\}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}.\ (\mathtt{w} = \mathtt{w1}\,\mathtt{w2}) \wedge \\
\qquad\qquad (((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w1}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\qquad\qquad\ \vee \\
\qquad\qquad\ ((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r2}) \wedge (\mathtt{M},\ \mathtt{w1}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r1})))) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r[*]}) = \\
\quad \exists \mathtt{wlist}.\ (\mathtt{w} = \mathtt{Concat}\ \mathtt{wlist}) \wedge \mathtt{Every}\ (\lambda \mathtt{w}.\ (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r}))\ \mathtt{wlist}) \\
\wedge \\
((\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r@c1}) = \\
\quad (\mathtt{M},\ \mathtt{w}\ \stackrel{\mathtt{c1}}{\models}\ \mathtt{r}))
\end{array}
$$

4.5 Foundation Language

We define a semantic function `F_SEM` such that `F_SEM M` π `c f` means FL formula `f` is true of path π if the current clock is `c`. The cases for weak (`c`) and strong (`c!`) clocks are considered separately.

We write $(\mathtt{M},\ \pi\ \stackrel{\mathtt{c}}{\models}\ \mathtt{r})$ for `F_SEM M` π `c f` and use the following two definitions:

$$\mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \quad = (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(0,\mathtt{i})}))\ \stackrel{\mathrm{T}}{\models}\ \neg\mathtt{c[*];c})$$

$$\mathtt{NextRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ (\mathtt{i},\mathtt{j}) = (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \stackrel{\mathrm{T}}{\models}\ \neg\mathtt{c[*];c})$$

The semantic clauses are then:

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{b}) = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \wedge (\mathtt{M},\ \mathtt{L_M}(\pi_{\mathtt{i}}) \models \mathtt{b})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \neg\mathtt{f}) = \\
\quad \neg(\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1} \wedge \mathtt{f2}) = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \wedge \\
\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1}) \quad \wedge \\
\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f2})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{X!\,f}) = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \wedge \\
\qquad\qquad \exists \mathtt{j} \in \mathtt{pl}\,\pi. \\
\qquad\qquad\quad \mathtt{i < j} \wedge \mathtt{NextRise\ M}\ \pi\ \mathtt{c\ (i+1,j)} \wedge (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{[f1\ U\ f2]}) = \\
\quad \exists \mathtt{i\ k} \in \mathtt{pl}\,\pi.\ \mathtt{k} \geq \mathtt{i} \qquad\qquad\qquad \wedge \\
\qquad\qquad\quad \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \qquad \wedge \\
\qquad\qquad\quad (\mathtt{M},\ \pi^{\mathtt{k}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c}) \qquad\quad \wedge \\
\qquad\qquad\quad (\mathtt{M},\ \pi^{\mathtt{k}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f2}) \qquad \wedge \\
\qquad\qquad\quad \forall \mathtt{j} \in \mathtt{pl}\,\pi. \\
\qquad\qquad\qquad \mathtt{i} \leq \mathtt{j} \wedge \mathtt{j} < \mathtt{k} \wedge \\
\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c}) \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \{\mathtt{r}\}(\mathtt{f})) = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \wedge \\
\qquad\qquad \forall \mathtt{j} \in \mathtt{pl}\,\pi.\ \mathtt{i} \leq \mathtt{j} \wedge (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \overset{\mathtt{c}}{\models}\ \mathtt{r}) \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}\!>}\{\mathtt{r2}\}\mathtt{!}) = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \wedge \\
\qquad\qquad \forall \mathtt{j} \in \mathtt{pl}\,\pi. \\
\qquad\qquad\qquad \mathtt{i} \leq \mathtt{j} \wedge (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \overset{\mathtt{c}}{\models}\ \mathtt{r1}) \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \exists \mathtt{k} \in \mathtt{pl}\,\pi.\ \mathtt{j} \leq \mathtt{k} \wedge (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\ \overset{\mathtt{c}}{\models}\ \mathtt{r2})) \\
\wedge
\end{array}
$$

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}>}\{\mathtt{r2}\})\ = \\
\ \ (\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}>}\{\mathtt{r2}\}\mathtt{!}) \\
\ \ \vee \\
\ \ ((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}>}\{\mathtt{r2}\})\ \wedge \\
\quad \forall \mathtt{j} \in \mathtt{pl}\,\pi.\ \exists \mathtt{k} \in \mathtt{pl}\,\pi.\ \mathtt{j}\ \leq\ \mathtt{k}\ \wedge\ \mathtt{NextRise\ M}\ \pi\ \mathtt{c\ (j,k)))} \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f\ abort\ b})\ = \\
\quad \exists \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i}\ \wedge \\
\qquad\qquad\qquad ((\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f}\ ) \\
\qquad\qquad\qquad\ \vee \\
\qquad\qquad\qquad\ \exists \mathtt{j} \in \mathtt{pl}\,\pi.\ \mathtt{i}\ <\ \mathtt{j}\ \wedge \\
\qquad\qquad\qquad\quad \exists \pi\mathtt{'}. \\
\qquad\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c} \wedge \mathtt{b})\ \wedge\ (\mathtt{M},\ \pi^{(\mathtt{i},\mathtt{j}-1)}\pi'\ \overset{\mathtt{c!}}{\models}\ \mathtt{f))) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f@c1})\ = \\
\quad (\mathtt{M},\ \pi\ \overset{\mathtt{c1}}{\models}\ \mathtt{f)) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f@c1!})\ = \\
\quad (\mathtt{M},\ \pi\ \overset{\mathtt{c1!}}{\models}\ \mathtt{f)) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{b})\ = \\
\quad \forall \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i}\ \Rightarrow\ (\mathtt{M},\ \mathtt{L}_{\mathtt{M}}(\pi_{\mathtt{i}})\ \models\ \mathtt{b)) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \neg\mathtt{f})\ = \\
\quad \neg(\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f)) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{f1} \wedge \mathtt{f2})\ = \\
\quad\ \forall \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \\
\qquad\qquad\qquad\ \Rightarrow \\
\qquad\qquad\qquad\ ((\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f1})\ \wedge\ (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f2))) } \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{X!\,f})\ = \\
\quad \forall \mathtt{i} \in \mathtt{pl}\,\pi.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i}\ \Rightarrow \\
\qquad\qquad\qquad \exists \mathtt{j} \in \mathtt{pl}\,\pi. \\
\qquad\qquad\qquad\quad \mathtt{i}\ <\ \mathtt{j}\ \wedge\ \mathtt{NextRise\ M}\ \pi\ \mathtt{c\ (i+1,j)}\ \wedge\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f)) } \\
\wedge
\end{array}
$$

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{[f1\ U\ f2]})\ \mathtt{=} \\
\quad (\mathtt{M},\ \pi\ \models^{\mathtt{c!}}\ \mathtt{[f1\ U\ f2]}) \\
\quad\ \vee \\
\quad (\exists \mathtt{k} \in \mathtt{pl}\,\pi\,\mathtt{.} \\
\qquad \forall \mathtt{l} \in \mathtt{pl}\,\pi\,\mathtt{.} \\
\qquad\quad \mathtt{l}\ \geq\ \mathtt{k}\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{l}}\ \models^{\mathrm{T}}\ \neg\mathtt{c}) \\
\qquad\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad\qquad \forall \mathtt{j} \in \mathtt{pl}\,\pi\,\mathtt{.} \\
\qquad\qquad\qquad\qquad\quad \mathtt{j}\ \leq\ \mathtt{k}\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models^{\mathrm{T}}\ \mathtt{c})\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models^{\mathtt{c}}\ \mathtt{f1}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \{\mathtt{r}\}(\mathtt{f}))\ \mathtt{=} \\
\quad \forall \mathtt{i} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i}\ \Rightarrow \\
\qquad\qquad\quad \forall \mathtt{j} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{i}\ \leq\ \mathtt{j}\ \wedge\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \models^{\mathtt{c}}\ \mathtt{r}) \\
\qquad\qquad\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\qquad\qquad\quad (\mathtt{M},\ \pi^{\mathtt{j}}\ \models^{\mathtt{c}}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \{\mathtt{r1}\}\mathtt{|\text{->}}\{\mathtt{r2}\}\mathtt{!})\ \mathtt{=} \\
\quad (\mathtt{M},\ \pi\ \models^{\mathtt{c!}}\ \{\mathtt{r1}\}\mathtt{|\text{->}}\{\mathtt{r2}\}\mathtt{!}) \\
\quad \vee \\
\quad ((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \{\mathtt{r1}\}\mathtt{|\text{->}}\{\mathtt{r2}\}) \\
\quad\ \wedge \\
\quad\ \exists \mathtt{k} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \forall \mathtt{l} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{l}\ \geq\ \mathtt{k}\ \Rightarrow\ (\mathtt{M},\ \pi^{\mathtt{l}}\ \models^{\mathrm{T}}\ \neg\mathtt{c}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \{\mathtt{r1}\}\mathtt{|\text{->}}\{\mathtt{r2}\})\ \mathtt{=} \\
\quad \forall \mathtt{i} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad \forall \mathtt{j} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{i}\ \leq\ \mathtt{j}\ \wedge\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \models^{\mathtt{c}}\ \mathtt{r1}) \\
\qquad\qquad\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\qquad\qquad\quad ((\exists \mathtt{k} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{j}\ \leq\ \mathtt{k}\ \wedge\ (\mathtt{M},\ \ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\ \models^{\mathtt{c}}\ \mathtt{r2})) \\
\qquad\qquad\qquad\qquad\quad\ \vee \\
\qquad\qquad\qquad\qquad\quad\ \forall \mathtt{k} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{j}\ \leq\ \mathtt{k}\ \Rightarrow\ \exists \mathtt{w}\mathtt{.}\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})})\,\mathtt{w})\ \models^{\mathtt{c}}\ \mathtt{r2}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{f\ abort\ b})\ \mathtt{=} \\
\quad \forall \mathtt{i} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad ((\mathtt{M},\ \pi^{\mathtt{i}}\ \models^{\mathtt{c}}\ \mathtt{f}) \\
\qquad\qquad\quad\ \vee \\
\qquad\qquad\quad\ \exists \mathtt{j} \in \mathtt{pl}\,\pi\,\mathtt{.}\ \mathtt{i}\ <\ \mathtt{j}\ \wedge \\
\qquad\qquad\quad\quad \exists \pi\text{'} \in \mathtt{sim}\ \pi\mathtt{.}\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models^{\mathrm{T}}\ \mathtt{c} \wedge \mathtt{b})\ \wedge\ (\mathtt{M},\ \pi^{(\mathtt{i},\mathtt{j}-1)}\pi'\ \models^{\mathtt{c}}\ \mathtt{f}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{f@c1})\ \mathtt{=} \\
\quad (\mathtt{M},\ \pi\ \models^{\mathtt{c1}}\ \mathtt{f})) \\
\wedge
\end{array}
$$

$$((\texttt{M},\ \pi \overset{\texttt{c}}{\models} \texttt{f@c1!}) = (\texttt{M},\ \pi \overset{\texttt{c1!}}{\models} \texttt{f}))$$

This semantics of FL formulas differs from the one we originally transcribed from the Accellera submission document [6]. See the Appendix for the original semantics and Section 5 for a discussion of the differences between it and the current semantics in Section 4.5.

4.6 Optional Branching Extension

The semantic function `O_SEM` is defined so that `O_SEM M s f` is true iff `f` is true of `M` at state `s`.

The semantics here differs from the simpler one in Section 3.3 in that it handles both finite and infinite paths.

Write $(\texttt{M},\ \texttt{s} \models \texttt{f})$ for `O_SEM M s f`, and then the semantics of the OBE is defined by:

$$
\begin{array}{l}
((\texttt{M},\ \texttt{s} \models \texttt{b}) = (\texttt{M},\ \texttt{L}_\texttt{M}(\texttt{s}) \models \texttt{b})) \\
\wedge \\
((\texttt{M},\ \texttt{s} \models \neg\texttt{f}) = \neg((\texttt{M},\ \texttt{s} \models \texttt{f}))) \\
\wedge \\
((\texttt{M},\ \texttt{s} \models \texttt{f1} \wedge \texttt{f2}) = (\texttt{M},\ \texttt{s} \models \texttt{f1}) \wedge (\texttt{M},\ \texttt{s} \models \texttt{f2})) \\
\wedge \\
((\texttt{M},\ \texttt{s} \models \texttt{EX f}) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge \texttt{pl}\ \pi\ \texttt{1} \wedge (\pi_0 = \texttt{s}) \wedge (\texttt{M},\ \pi_1 \models \texttt{f})) \\
\wedge \\
((\texttt{M},\ \texttt{s} \models \texttt{[f1 U f2]}) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge (\pi_0 = \texttt{s}) \\
\qquad \wedge \\
\qquad \exists\texttt{k} \in \texttt{pl}\,\pi. \\
\qquad\quad (\texttt{M},\ \pi_\texttt{k} \models \texttt{f2}) \wedge \forall\texttt{j} \in \texttt{pl}\,\pi.\ \texttt{j} < \texttt{k} \Rightarrow (\texttt{M},\ \pi_\texttt{j} \models \texttt{f1})) \\
\wedge \\
((\texttt{M},\ \texttt{s} \models \texttt{EG f}) = \\
\quad \exists\pi.\ \texttt{Path M}\ \pi \wedge (\pi_0 = \texttt{s}) \wedge \forall\texttt{j} \in \texttt{pl}\,\pi.\ (\texttt{M},\ \pi_\texttt{j} \models \texttt{f}))
\end{array}
$$

5 Progress on analysing the semantics

We have established a number of properties of the semantics using the HOL system. Some of these went through first time without any problems, but others revealed bugs both in the *Sugar 2.0* semantics and original HOL representation of the semantics.

5.1 Properties of `FirstRise` and `NextRise`

Recall the definitions:

$$\texttt{FirstRise M } \pi \texttt{ c i} = (\texttt{M}, (\hat{\texttt{L}}_{\texttt{M}} (\pi^{(0,\texttt{i})})) \overset{\mathrm{T}}{\models} \neg\texttt{c[*];c})$$
$$\texttt{NextRise M } \pi \texttt{ c (i,j)} = (\texttt{M}, (\hat{\texttt{L}}_{\texttt{M}} (\pi^{(\texttt{i},\texttt{j})})) \overset{\mathrm{T}}{\models} \neg\texttt{c[*];c})$$

We have proved that the definitions of `FirstRise` and `NextRise` give them the correct meaning, namely `FirstRise` M π `c i` is true iff `i` is the time of the first rising edge of `c`, and `NextRise` M π `c (i,j)` is true iff `j` is the time of the first rising edge of `c` after `i`.

$$\vdash \texttt{FirstRise M } \pi \texttt{ c i} = (\forall \texttt{j}.\ \texttt{j} < \texttt{i} \Rightarrow \neg(\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{j}}) \models \texttt{c})) \wedge (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}}) \models \texttt{c})$$

$$\vdash \texttt{i} \leq \texttt{j} \Rightarrow (\texttt{NextRise M } \pi \texttt{ c (i,j)} = (\forall \texttt{k}.\ \texttt{i} \leq \texttt{k} \wedge \texttt{k} < \texttt{j} \Rightarrow \neg(\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{k}}) \models \texttt{c})) \wedge (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{j}}) \models \texttt{c}))$$

The proof of these were essentially routine, though quite a bit more tricky than expected. Immediate corollaries are

$$\vdash \texttt{FirstRise M } \pi \texttt{ T i} = (\texttt{i} = 0)$$
$$\vdash \texttt{i} \leq \texttt{j} \Rightarrow (\texttt{NextRise M } \pi \texttt{ T (i,j)} = (\texttt{i} = \texttt{j}))$$

5.2 Relating the clocked and unclocked semantics

If we define `ClockFree r` to mean that `r` contains no clocking constructs (a simple recursion over the syntax of SEREs), then clocking with T is equivalent to the unclocked SERE semantics.

$$\vdash \forall \texttt{r}.\ \texttt{ClockFree r} \Rightarrow ((\texttt{M, w} \overset{\mathrm{T}}{\models} \texttt{r}) = (\texttt{M, w} \models \texttt{r}))$$

The proof of this is an easy structural induction, and shows that the semantics in Section 4.4 collapses to that in Section 3.1 when the clock is T.

We tried to prove a similar result for FL formulas, but this turned out to be impossible. The reason is that the proof required first showing

$$\forall \texttt{f}\ \pi.\ (\texttt{M}, \pi \overset{\mathrm{T}}{\models} \texttt{f}) = (\texttt{M}, \pi \overset{\mathrm{T}\texttt{!}}{\models} \texttt{f}))$$

However, the original semantics had the following:

$$(\texttt{M}, \pi \overset{\texttt{c!}}{\models} \texttt{b}) = \exists \texttt{i}.\ \texttt{FirstRise M } \pi \texttt{ c i} \wedge (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}}) \models \texttt{b})$$

$$(\texttt{M}, \pi \overset{\texttt{c}}{\models} \texttt{b}) = \exists \texttt{i}.\ \texttt{FirstRise M } \pi \texttt{ c i} \Rightarrow (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}}) \models \texttt{b})$$

Instantiating `c` to T and using the corollary about `FirstRise` yields

$$(\texttt{M}, \pi \overset{\mathrm{T}\texttt{!}}{\models} \texttt{b}) = \exists \texttt{i}.\ (\texttt{i=0}) \wedge (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}}) \models \texttt{b})$$

$$(\texttt{M}, \pi \overset{\mathrm{T}}{\models} \texttt{b}) = \exists \texttt{i}.\ (\texttt{i=0}) \Rightarrow (\texttt{M}, \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}}) \models \texttt{b})$$

With this, clearly $(\mathtt{M},\ \pi\ \overset{\mathrm{T}}{\models}\ \mathtt{b})$ is not equal to $(\mathtt{M},\ \pi\ \overset{\mathrm{T!}}{\models}\ \mathtt{b}))$. The solution, suggested by Cindy Eisner, is to replace the weak semantics by

$$(\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{b}) \ = \forall \mathtt{i}.\ \mathtt{FirstRise\ M}\ \pi\ \mathtt{c\ i} \Rightarrow (\mathtt{M},\ \mathtt{L_M}(\pi_{\mathtt{i}})\ \models \mathtt{b})$$

so that we get

$$(\mathtt{M},\ \pi\ \overset{\mathrm{T!}}{\models}\ \mathtt{b}) = \exists \mathtt{i}.\ (\mathtt{i=0}) \wedge (\mathtt{M},\ \mathtt{L_M}(\pi_{\mathtt{i}})\ \models \mathtt{b})$$

$$(\mathtt{M},\ \pi\ \overset{\mathrm{T}}{\models}\ \mathtt{b}) \ = \forall \mathtt{i}.\ (\mathtt{i=0}) \Rightarrow (\mathtt{M},\ \mathtt{L_M}(\pi_{\mathtt{i}})\ \models \mathtt{b})$$

which makes $(\mathtt{M},\ \pi\ \overset{\mathrm{T}}{\models}\ \mathtt{b})$ equal to $(\mathtt{M},\ \pi\ \overset{\mathrm{T!}}{\models}\ \mathtt{b}))$. The same change of $\exists$ to $\forall$ is also needed for the semantics of weak clocking for $\mathtt{f1} \wedge \mathtt{f2}$, `X! f`, $\{\mathtt{r}\}(\mathtt{f})$, $\{\mathtt{r1}\}\mathtt{|\text{-}\!>}\{\mathtt{r2}\}$ and `f abort b`. With these changes, a structural induction proves:

$$\vdash\ \forall \mathtt{f}\ \pi.\ (\mathtt{M},\ \pi\ \overset{\mathrm{T}}{\models}\ \mathtt{f})\ =\ (\mathtt{M},\ \pi\ \overset{\mathrm{T!}}{\models}\ \mathtt{f}))$$

However, we were still unable to prove

$$\vdash\ \forall \mathtt{f}.\ \mathtt{ClockFree\ f}\ \Rightarrow\ ((\mathtt{M},\ \mathtt{w}\ \overset{\mathrm{T}}{\models}\ \mathtt{f})\ =\ (\mathtt{M},\ \mathtt{w}\ \models\ \mathtt{f}))$$

where here `ClockFree f` means that `f` contains no clocked FL formulas or SEREs, the semantics on the left of the equation is the one in Section 4.5 and the semantics on the right of the equation is the one in Section 4.2. The proof attempt failed because the unclocked semantics for `[f1 U f2]` had a path length check, but the strongly clocked semantics didn't. After restricting the quantification of `k` in the strongly clocked semantics to values satisfying `pl` π the proof went through.

5.3 Further analysis

The original semantics specifies that some of the quantifications over the variables `i`, `j`, `k` etc. be restricted to range over values the are smaller than the length of the current path π (we represent this using `pl` π). Our initial attempts to relate the clocked and unclocked semantics needed additional quantifier restrictions to be added, as discussed at the end of Section 5.2 above. However, during email discussions with the *Sugar 2.0* designers it became clear that in fact all quantifications should be restricted, for otherwise the semantics would rely on the HOL logic's default interpretions of terms like π^j when π is finite and $j \geq \mathtt{length}\ \pi$.[11] With HOL's default interpretation of 'meaningless' terms, it is unclear whether the semantics accurately reflects the designers intentions.

[11] The logical treatment of 'undefined' terms like 1/0 or `hd[]` has been much discussed. HOL uses a simple and consistent approach based on Hilbert's ε-operator. Other approaches include 'free logics' (i.e. logics with non-denoting terms) and three-valued logics in which formulas can evaluate to *true*, *false* and *undefined*.

Thus the semantics was modified so that all quantifications are suitably restricted. In addition, and in the same spirit, we added the requirement that all terms $\pi^{(m,n)}$ occured in a context where $m \leq n$, so that the arbitrary value of $\pi^{(m,n)}$ when $m > n$ was not invoked. Unfortunately these changes broke the existing proof of

$$\vdash\ \forall \mathtt{f}\ \pi.\ (\mathtt{M},\ \pi \overset{\mathtt{T}}{\models} \mathtt{f})\ =\ (\mathtt{M},\ \pi \overset{\mathtt{T!}}{\models} \mathtt{f}))$$

and hence the proof relating the clocked and unclocked semantics.

Further discussion with the *Sugar 2.0* designers at first suggested that in fact this property should not be expected to hold after all. What we thought had happened was that our semantics, following the Accellera submission semantics, was imprecise about certain details, and this imprecision conspired to make the property true. When we sharpened the semantics (to eliminate meaningless terms), the property became false. However, in the end it turned out that there was just a bug in the semantics: "$\mathtt{l} > \mathtt{k}$" should be "$\mathtt{l} \geq \mathtt{k}$" in the weakly clocked semantics of until and strong suffix implication, and when this change was made the proof of the above property, and the equivalence between the unclocked and true-clocked semantics, went through. However, just as we thought everything was sorted out, the *Sugar 2.0* designers announced they had discovered a bug and pointed out that we should not have been able to prove what we had without a tweak. Careful analysis showed that a discrepancy between the semantics in this paper and what was defined in HOL had crept in. The bug in the HOL semantics allowed a proof to succeed when it shouldn't have. After fixing the discrepancy the proofs failed, as they should, and after the correct fix was made they went through.

6 Conclusions

It was quite straightforward to use the informal semantics in the *Sugar 2.0* documentation to create a deep embedding of the whole *Sugar 2.0* kernel. Attempting to prove some simple 'sanity checking' lemmas with a proof assistant quickly revealed bugs in the translated semantics (and possibly in the original). Further probing revealed more bugs.

It is hoped that the semantics in Section 4 is correct, but until further properties are proved we cannot be sure, and the experience so far suggests caution!

7 Acknowledgements

Cindy Eisner and Dana Fisman patiently answered numerous email questions in great detail. They also supplied valuable comments and corrections to an earlier version of this paper, and suggested ways of modifying the HOL semantics to get the proofs described in Section 5 to go through.

References

1. I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic Sugar. In G. Berry, H. Comon, and A. Finkel, editors, *Proc.* 13^{th} *International Conference on Computer Aided Verification (CAV)*, LNCS 2102. Springer-Verlag, 2001.
2. M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: a theorem-proving environment for higher-order logic.* Cambridge University Press, 1993.
3. J. Halpern, Z. Manna, and B. Moszkowski. A hardware semantics based on temporal intervals. In J. Diaz, editor, *Proceedings of the 10-th International Colloquium on Automata, Languages and Programming*, volume 154 of *LNCS*, pages 278–291. Springer Verlag, 1983.
4. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
5. S. Rajan, N. Shankar, and M.K. Srivas. An integration of model-checking with automated proof checking. In Pierre Wolper, editor, *Computer-Aided Verification, CAV '95*, volume 939 of *Lecture Notes in Computer Science*, pages 84–97, Liege, Belgium, June 1995. Springer-Verlag.
6. `www.haifa.il.ibm.com/projects/verification/sugar/literature.html`.

APPENDIX: Initial HOL semantics

This appendix consists of a typeset version of our initial transcription into higher order logic of the semantics in the Sugar documentation

http://www.haifa.il.ibm.com/projects/verification/sugar/literature.html

This semantics turned out to be inadequate for the reasons discussed in Section 5.

8 Boolean expressions

$$
\begin{array}{lll}
((\mathtt{M},\ \mathtt{l} \models \mathtt{p}) & = p \in \mathtt{P}_M \wedge p \in \mathtt{l}) & \wedge \\
((\mathtt{M},\ \mathtt{l} \models \mathtt{T}) & = \mathtt{T}) & \wedge \\
((\mathtt{M},\ \mathtt{l} \models \neg\mathtt{b}) & = \neg(\mathtt{M},\ \mathtt{l} \models \mathtt{b})) & \wedge \\
((\mathtt{M},\ \mathtt{l} \models \mathtt{b1} \wedge \mathtt{b2}) & = (\mathtt{M},\ \mathtt{l} \models \mathtt{b1}) \wedge (\mathtt{M},\ \mathtt{l} \models \mathtt{b2})) &
\end{array}
$$

9 Sugar Extended Regular Expressions

If `wlist` is a list of lists then `Concat wlist` is the concatenation of the lists in `wlist` and if P is some predicate then `Every` P `wlist` means that $P(\mathtt{w})$ holds for every `w` in `wlist`.

$$
\begin{array}{l}
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{b}) = \\
\quad \exists \mathtt{n}.\ \mathtt{n} \geq 1 \hfill \wedge \\
\qquad (\text{length } \mathtt{w} = \mathtt{n}) \hfill \wedge \\
\qquad (\forall \mathtt{i}.\ 1 \leq \mathtt{i} \wedge \mathtt{i} < \mathtt{n} \Rightarrow (\mathtt{M},\ \mathtt{w}_{\mathtt{i}-1} \models \neg\mathtt{c}) \wedge \\
\qquad (\mathtt{M},\ \mathtt{w}_{\mathtt{n}-1} \models \mathtt{c} \wedge \mathtt{b})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r1;r2}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}.\ (\mathtt{w} = \mathtt{w1}\,\mathtt{w2}) \wedge (\mathtt{M},\ \mathtt{w1} \overset{\mathtt{c}}{\models} \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w2} \overset{\mathtt{c}}{\models} \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r1:r2}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}\ \mathtt{l}.\ (\mathtt{w} = \mathtt{w1}\,[\mathtt{l}]\,\mathtt{w2}) \wedge \\
\qquad\qquad (\mathtt{M},\ (\mathtt{w1}\,[\mathtt{l}]) \overset{\mathtt{c}}{\models} \mathtt{r1}) \wedge (\mathtt{M},\ ([\mathtt{l}]\,\mathtt{w2}) \overset{\mathtt{c}}{\models} \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \{\mathtt{r1}\}|\{\mathtt{r2}\}) = \\
\quad (\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r1}) \vee (\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \{\mathtt{r1}\}\&\&\{\mathtt{r2}\}) = \\
\quad (\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \{\mathtt{r1}\}\&\{\mathtt{r2}\}) = \\
\quad \exists \mathtt{w1}\ \mathtt{w2}.\ (\mathtt{w} = \mathtt{w1}\,\mathtt{w2}) \wedge \\
\qquad\quad (((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r1}) \wedge (\mathtt{M},\ \mathtt{w1} \overset{\mathtt{c}}{\models} \mathtt{r2})) \\
\qquad\quad \vee \\
\qquad\quad ((\mathtt{M},\ \mathtt{w} \overset{\mathtt{c}}{\models} \mathtt{r2}) \wedge (\mathtt{M},\ \mathtt{w1} \overset{\mathtt{c}}{\models} \mathtt{r1}))))
\end{array}
$$

$\wedge$
$((\mathtt{M},\ \mathtt{w}\ \overset{\mathtt{c}}{\models}\ \mathtt{r[*]}) =$
$\quad \exists \mathtt{wlist}.\ (\mathtt{w} = \mathtt{Concat\ wlist}) \wedge \mathtt{Every}\ (\lambda \mathtt{w}.\ (\mathtt{M},\ \mathtt{w}\ \overset{\mathtt{c}}{\models}\ \mathtt{r}))\ \mathtt{wlist})$
$\wedge$
$((\mathtt{M},\ \mathtt{w}\ \overset{\mathtt{c}}{\models}\ \mathtt{r@c1}) =$
$\quad (\mathtt{M},\ \mathtt{w}\ \overset{\mathtt{c1}}{\models}\ \mathtt{r}))$

10 Foundation Language

The following two constants are defined:

$$\mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \quad = (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{0},\mathtt{i})}))\ \overset{\mathrm{T}}{\models}\ \neg\mathtt{c[*];c})$$
$$\mathtt{NextRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{(i,j)} = (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \overset{\mathrm{T}}{\models}\ \neg\mathtt{c[*];c})$$

The semantic clauses are then:

$((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{b}) =$
$\quad \exists \mathtt{i}.\ \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \wedge (\mathtt{M},\ \mathtt{L}_{\mathtt{M}}(\pi_{\mathtt{i}})\ \models\ \mathtt{b})$
$\wedge$
$((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \neg\mathtt{f}) =$
$\quad \neg((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{f}))$
$\wedge$
$((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1} \wedge \mathtt{f2}) =$
$\quad \exists \mathtt{i}.\ \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \wedge$
$\qquad (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1}) \quad \wedge$
$\qquad (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f2}))$
$\wedge$
$((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{X!}\ \mathtt{f}) =$
$\quad \exists \mathtt{i}.\ \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \qquad \wedge$
$\qquad (\text{finite}\ \pi \Rightarrow \mathtt{i} < \text{length}\ \pi\ \mathtt{-}\ \mathtt{1}) \wedge$
$\qquad (\mathtt{M},\ \pi^{\mathtt{i+1}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f}))$
$\wedge$
$((\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{[f1\ U\ f2]}) =$
$\quad \exists \mathtt{i}\ \mathtt{k}.\ \mathtt{k} \geq \mathtt{i} \qquad \wedge$
$\qquad \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \quad \wedge$
$\qquad (\mathtt{M},\ \pi^{\mathtt{k}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c}) \qquad \wedge$
$\qquad (\mathtt{M},\ \pi^{\mathtt{k}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f2}) \qquad \wedge$
$\qquad \forall \mathtt{j}.\ \mathtt{i} \leq \mathtt{j} \wedge \mathtt{j} < \mathtt{k} \wedge$
$\qquad\quad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c})$
$\qquad\quad \Rightarrow$
$\qquad\quad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c!}}{\models}\ \mathtt{f1}))$
$\wedge$

$$
\begin{array}{l}
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \{\texttt{r}\}(\texttt{f})) = \\
\quad \exists \texttt{i}.\ \texttt{FirstRise M}\ \pi\ \texttt{c i} \wedge \\
\qquad\quad \forall \texttt{j}.\ (\texttt{M},\ (\hat{\texttt{L}}_{\texttt{M}}\ (\pi^{(\texttt{i},\texttt{j})}))\ \stackrel{\texttt{c}}{\models}\ \texttt{r}) \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad (\texttt{M},\ \pi^{\texttt{j}}\ \stackrel{\texttt{c!}}{\models}\ \texttt{f})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \{\texttt{r1}\}\texttt{|->}\{\texttt{r2}\}\texttt{!}) = \\
\quad \exists \texttt{i}.\ \texttt{FirstRise M}\ \pi\ \texttt{c i} \wedge \\
\qquad\quad \forall \texttt{j}.\ (\texttt{M},\ (\hat{\texttt{L}}_{\texttt{M}}\ (\pi^{(\texttt{i},\texttt{j})}))\ \stackrel{\texttt{c}}{\models}\ \texttt{r1}) \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad \exists \texttt{k}.\ (\texttt{M},\ (\hat{\texttt{L}}_{\texttt{M}}\ (\pi^{(\texttt{j},\texttt{k})}))\ \stackrel{\texttt{c}}{\models}\ \texttt{r2})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \{\texttt{r1}\}\texttt{|->}\{\texttt{r2}\}) = \\
\ \ (\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \{\texttt{r1}\}\texttt{|->}\{\texttt{r2}\}\texttt{!}) \\
\quad \vee \\
\ \ ((\texttt{M},\ \pi \ \stackrel{\texttt{c}}{\models}\ \{\texttt{r1}\}\texttt{|->}\{\texttt{r2}\}) \\
\quad \wedge \\
\quad \forall \texttt{j}.\ (\mathrm{finite}\ \pi\ \Rightarrow\ \texttt{j}\ <\ \mathrm{length}\ \pi) \\
\qquad\quad \Rightarrow \\
\qquad\quad \exists \texttt{k}.\ \texttt{NextRise M}\ \pi\ \texttt{c (j,k)})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \texttt{f abort b}) = \\
\quad \exists \texttt{i}.\ \texttt{FirstRise M}\ \pi\ \texttt{c i} \wedge \\
\qquad\quad ((\texttt{M},\ \pi^{\texttt{i}}\ \stackrel{\texttt{c!}}{\models}\ \texttt{f}\ ) \\
\qquad\qquad \vee \\
\qquad\quad\ \ \exists \texttt{j}\ \pi\texttt{'}. \\
\qquad\qquad\quad (\texttt{M},\ \pi^{\texttt{j}}\ \stackrel{\mathrm{T}}{\models}\ \texttt{c} \wedge \texttt{b})\ \wedge \\
\qquad\qquad\quad (\texttt{M},\ \pi^{(\texttt{i},\texttt{j}-1)}\pi'\ \stackrel{\texttt{c!}}{\models}\ \texttt{f}))) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \texttt{f@c1}) = \\
\quad (\texttt{M},\ \pi \ \stackrel{\texttt{c1}}{\models}\ \texttt{f})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \texttt{f@c1!}) = \\
\quad (\texttt{M},\ \pi \ \stackrel{\texttt{c1!}}{\models}\ \texttt{f})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c}}{\models}\ \texttt{b}) = \\
\quad \exists \texttt{i}.\ \texttt{FirstRise M}\ \pi\ \texttt{c i} \\
\qquad\quad \Rightarrow \\
\qquad\quad (\texttt{M},\ \texttt{L}_{\texttt{M}}(\pi_{\texttt{i}})\ \models\ \texttt{b})) \\
\wedge \\
((\texttt{M},\ \pi \ \stackrel{\texttt{c}}{\models}\ \neg\texttt{f}) = \\
\quad \neg((\texttt{M},\ \pi \ \stackrel{\texttt{c!}}{\models}\ \texttt{f})))
\end{array}
$$

$$
\begin{array}{l}
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{f1} \wedge \mathtt{f2})\ \mathtt{=} \\
\quad \exists \mathtt{i.\ FirstRise\ M}\ \pi\ \mathtt{c\ i} \\
\qquad\quad \Rightarrow \\
\qquad\quad ((\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f1}) \\
\qquad\qquad \wedge \\
\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{i}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f2}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{X!\,f})\ \mathtt{=} \\
\quad \exists \mathtt{i.\ (FirstRise\ M}\ \pi\ \mathtt{c\ i} \\
\qquad\qquad \wedge \\
\qquad\qquad (\text{finite}\ \pi\ \Rightarrow\ \mathtt{i}\ <\ \text{length}\ \pi\ \mathtt{-\ 1})) \\
\qquad\qquad \Rightarrow \\
\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{i+1}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \mathtt{[f1\ U\ f2]})\ \mathtt{=} \\
\quad (\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \mathtt{[f1\ U\ f2]}) \\
\quad \vee \\
\quad (\exists \mathtt{k.}\ \forall \mathtt{l.\ l}\ >\ \mathtt{k} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{l}}\ \overset{\mathrm{T}}{\models}\ \neg\mathtt{c}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad \forall \mathtt{j.\ j}\ \leq\ \mathtt{k} \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathrm{T}}{\models}\ \mathtt{c}) \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f1}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \{\mathtt{r}\}(\mathtt{f}))\ \mathtt{=} \\
\quad \exists \mathtt{i.\ FirstRise\ M}\ \pi\ \mathtt{c\ i}\ \Rightarrow \\
\qquad\quad \forall \mathtt{j.\ (M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \overset{\mathtt{c}}{\models}\ \mathtt{r}) \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad (\mathtt{M},\ \pi^{\mathtt{j}}\ \overset{\mathtt{c}}{\models}\ \mathtt{f})) \\
\wedge \\
\\
((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}\!>}\{\mathtt{r2}\}\mathtt{!})\ \mathtt{=} \\
\quad (\mathtt{M},\ \pi\ \overset{\mathtt{c!}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}\!>}\{\mathtt{r2}\}\mathtt{!}) \\
\quad \vee \\
\quad ((\mathtt{M},\ \pi\ \overset{\mathtt{c}}{\models}\ \{\mathtt{r1}\}\mathtt{|\text{-}\!>}\{\mathtt{r2}\}) \\
\quad\ \ \wedge \\
\quad\ \ \exists \mathtt{k.}\ \forall \mathtt{l.\ l}\ >\ \mathtt{k} \\
\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad (\mathtt{M},\ \pi^{\mathtt{l}}\ \overset{\mathrm{T}}{\models}\ \neg\mathtt{c}))) \\
\wedge
\end{array}
$$

$$
\begin{array}{l}
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \{\mathtt{r1}\}\texttt{|->}\{\mathtt{r2}\})\ = \\
\quad \exists \mathtt{i}.\ \ \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \\
\qquad\quad \Rightarrow \\
\qquad\quad \forall \mathtt{j}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{i},\mathtt{j})}))\ \models^{\mathtt{c}}\ \mathtt{r1}) \\
\qquad\qquad\quad \Rightarrow \\
\qquad\qquad\quad ((\exists \mathtt{k}.\ (\mathtt{M},\ \ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})}))\ \models^{\mathtt{c}}\ \mathtt{r2})) \\
\qquad\qquad\qquad \vee \\
\qquad\qquad\qquad \forall \mathtt{k}.\ (\text{finite}\ \pi\ \Rightarrow\ \mathtt{k}\ <\ \text{length}\ \pi) \\
\qquad\qquad\qquad\qquad \Rightarrow \\
\qquad\qquad\qquad\qquad \exists \mathtt{w}.\ (\mathtt{M},\ (\hat{\mathtt{L}}_{\mathtt{M}}\ (\pi^{(\mathtt{j},\mathtt{k})})\,\mathtt{w})\ \models^{\mathtt{c}}\ \mathtt{r2}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{f\ abort\ b})\ = \\
\quad \exists \mathtt{i}.\ \mathtt{FirstRise}\ \mathtt{M}\ \pi\ \mathtt{c}\ \mathtt{i} \\
\qquad \Rightarrow \\
\qquad ((\mathtt{M},\ \pi^{\mathtt{i}}\ \models^{\mathtt{c}}\ \mathtt{f}) \\
\qquad\ \vee \\
\qquad\ \exists \mathtt{j}\ \pi'.\ (\mathtt{M},\ \pi^{\mathtt{j}}\ \models^{\mathrm{T}}\ \mathtt{c} \wedge \mathtt{b}) \\
\qquad\qquad\qquad \wedge \\
\qquad\qquad\qquad (\mathtt{M},\ \pi^{(\mathtt{i},\mathtt{j}-1)}\,\pi'\ \models^{\mathtt{c}}\ \mathtt{f}))) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{f@c1})\ = \\
\quad (\mathtt{M},\ \pi\ \models^{\mathtt{c1}}\ \mathtt{f})) \\
\wedge \\
((\mathtt{M},\ \pi\ \models^{\mathtt{c}}\ \mathtt{f@c1!})\ = \\
\quad (\mathtt{M},\ \pi\ \models^{\mathtt{c1!}}\ \mathtt{f}))
\end{array}
$$

11 Optional Branching Extension

The following predicate is defined

$$\mathtt{Path}\ \mathtt{M}\ \pi\ =\ \forall \mathtt{n}.\ \mathtt{R}_{M}(\pi_n,\ \pi_{n+1})$$

The semantic clauses are then:

$$
\begin{array}{l}
((\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{b})\ =\ (\mathtt{M},\ \mathtt{L}_{\mathtt{M}}(\mathtt{s})\ \models\ \mathtt{b})) \\
\wedge \\
((\mathtt{M},\ \mathtt{s}\ \models\ \neg\mathtt{f})\ =\ \neg((\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{f}))) \\
\wedge \\
((\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{f1} \wedge \mathtt{f2})\ =\ (\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{f1})\ \wedge\ (\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{f2})) \\
\wedge \\
((\mathtt{M},\ \mathtt{s}\ \models\ \mathtt{EX}\ \mathtt{f})\ = \\
\quad \exists \pi.\ \mathtt{Path}\ \mathtt{M}\ \pi \qquad\qquad\qquad\qquad \wedge \\
\qquad\ (\text{finite}\ \pi\ \Rightarrow\ \mathtt{1}\ <\ \text{length}\ \pi)\ \wedge \\
\qquad\ (\pi_0\ =\ \mathtt{s})\ \wedge\ (\mathtt{M},\ \pi_1\ \models\ \mathtt{f}))
\end{array}
$$

$$
\begin{array}{l}
\wedge \\
((\mathtt{M},\ \mathtt{s} \models \mathtt{[f1\ U\ f2]}) = \\
\quad \exists\pi.\ \mathtt{Path\ M}\ \pi \qquad\qquad\qquad\qquad\qquad\qquad \wedge \\
\qquad\quad (\pi_0\ \mathtt{=\ s}) \qquad\qquad\qquad\qquad\qquad\quad \wedge \\
\qquad\quad \exists\mathtt{k}.\ (\mathrm{finite}\ \pi \Rightarrow \mathtt{k} < \mathrm{length}\ \pi) \ \wedge \\
\qquad\qquad\quad (\mathtt{M},\ \pi_{\mathtt{k}} \models \mathtt{f2}) \qquad\qquad\qquad \wedge \\
\qquad\qquad\quad \forall\mathtt{j}.\ \mathtt{j} < \mathtt{k} \Rightarrow (\mathtt{M},\ \pi_{\mathtt{j}} \models \mathtt{f1})) \\
\wedge \\
((\mathtt{M},\ \mathtt{s} \models \mathtt{EG\ f}) = \\
\quad \exists\pi.\ \mathtt{Path\ M}\ \pi\ \wedge \\
\qquad\quad (\pi_0\ \mathtt{=\ s})\ \ \wedge \\
\qquad\quad \forall\mathtt{j}.\ (\mathrm{finite}\ \pi \Rightarrow \mathtt{j} < \mathrm{length}\ \pi) \Rightarrow (\mathtt{M},\ \pi_{\mathtt{j}} \models \mathtt{f}))
\end{array}
$$