

Proposal for Sugar 2.0 Temporal Layer Formal Syntax and Semantics “SEM_1”

Cindy Eisner¹ Dana Fisman^{1,2} Michael J.C. Gordon³
John Havlicek⁴ Anthony McIsaac⁵ David Van Campenhout⁶

¹ IBM Haifa Research Lab ² Weizmann Institute of Science ³ University of Cambridge
⁴ Motorola, Inc. ⁵ STMicroelectronics, Ltd. ⁶ Verisity Design, Inc.

December 10, 2002

1 Syntax

Boolean expression syntax varies according to the Sugar flavor used. The formal syntax definition uses the complete set $\{\neg, \wedge\}$, and semantics are given here only to these two operators. Semantics of any other boolean expression follow directly from these.

Definition 1 (Boolean expression).

- Every atomic proposition is a boolean expression.
- If b , b_1 , and b_2 are boolean expressions, then so are the following:
 - (b)
 - $\neg b$
 - $b_1 \wedge b_2$

Definition 2 (Sugar Extended Regular Expressions (SEREs)).

- Every boolean expression is a SERE.
- If r , r_1 , and r_2 are SEREs, and clk is a boolean expression, then the following are SEREs:
 - $\{r\}$
 - $r_1 ; r_2$
 - $r_1 : r_2$
 - $\{r_1\} \mid \{r_2\}$
 - $\{r_1\} \&\& \{r_2\}$
 - $r[*]$
 - $r@clk$

Definition 3 (Formulas of the Sugar Foundation Language (FL)).

- Every boolean expression is a Sugar FL formula.
- If b and clk are boolean expressions, f , f_1 , and f_2 are Sugar FL formulas and r , r_1 , and r_2 are SEREs, then the following are Sugar FL formulas:
 - (f)

- $\neg f$
- $f_1 \wedge f_2$
- $X! f$
- $[f_1 U f_2]$
- $\{r\}(f)$
- $\{r_1\} \mapsto \{r_2\}!$
- $\{r_1\} \mapsto \{r_2\}$
- $f \text{ abort } b$
- $f@clk$
- $f@clk!$

In Section 3, we show additional operators which provide syntactic sugaring to those described above.

Definition 4 (Formulas of the Optional Branching Extension (OBE)).

- *Every boolean expression is an OBE formula.*
- *If f , f_1 , and f_2 are OBE formulas, then so are the following:*
 - (f)
 - $\neg f$
 - $f_1 \wedge f_2$
 - EXf
 - $E[f_1 U f_2]$
 - EGf

Additional OBE operators are derived from these as follows ¹:

- $f_1 \vee f_2 = \neg(\neg f_1 \wedge \neg f_2)$
- $f_1 \rightarrow f_2 = \neg f_1 \vee f_2$
- $f_1 \leftrightarrow f_2 = (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$
- $EFf = E[\top U f]$
- $AXf = \neg EX\neg f$
- $A[f_1 U f_2] = \neg(E[\neg f_2 U (\neg f_1 \wedge \neg f_2)] \vee EG\neg f_2)$
- $AGf = \neg E[\top U \neg f]$
- $AFf = A[\top U f]$

Definition 5 (Sugar Formulas).

- *Every Sugar FL formula is a Sugar formula.*
- *Every OBE formula is a Sugar formula.*

¹ Where $\top = p \vee \neg p$ for some $p \in P$.

2 Semantics

The semantics of a Sugar formula are defined with respect to a *model* M . A model is a quintuple (S, S_0, R, P, L) , where S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $R \subseteq S \times S$ is the transition relation, P is a non-empty set of atomic propositions, and L is the valuation, a function $L : S \rightarrow 2^P$, mapping each state with a set of atomic propositions valid in that state.

A *path* π is a finite (or infinite) sequence of states $\pi = (\pi_0, \pi_1, \pi_2, \dots, \pi_n)$ (or $\pi = (\pi_0, \pi_1, \pi_2, \dots)$). A *computation path* π of a model M is a finite (or infinite) path π such that for every $i < n$, $R(\pi_i, \pi_{i+1})$ and for no s , $R(\pi_n, s)$ (or such that for every i , $R(\pi_i, \pi_{i+1})$). Given a finite (or infinite) path π , we define \hat{L} , an extension of the valuation function L from states to paths as follows: $\hat{L}(\pi) = L(\pi_0)L(\pi_1)\dots L(\pi_n)$ (or $\hat{L}(\pi) = L(\pi_0)L(\pi_1)\dots$). Thus we have a mapping from states in M to letters of 2^P , and from finite (or infinite) sequences of states in M to finite (or infinite) words over 2^P .

We will denote a letter from 2^P by ℓ , and a finite or infinite word from 2^P by ω . We denote the length of word ω as $|\omega|$. A finite word $\omega = \ell_0\ell_1\ell_2\dots\ell_n$ has length $n+1$, while an infinite word has length ∞ . We denote by ω^i the suffix of ω starting at ℓ_i . That is, $\omega^i = \ell_i\ell_{i+1}\dots\ell_n$ (or $\omega^i = \ell_i\ell_{i+1}\dots$). We denote by $\omega^{i,j}$ the finite sequence of letters starting from ℓ_i and ending in ℓ_j . That is, $\omega^{i,j} = \ell_i\ell_{i+1}\dots\ell_j$.

For readability, we first define the semantics of unlocked Sugar formulas (and SEREs) and only then the semantics of clocked Sugar formulas (and clocked SEREs). In fact, the semantics of unlocked Sugar formulas (and unlocked SEREs) can be obtained from the semantics of clocked Sugar formulas (and clocked SEREs) by replacing the clock context with \top ².

Semantics of boolean expressions

We define the semantics of boolean expressions over letters from the alphabet 2^P , thus a letter is a subset of the set of atomic propositions P . The notation $\ell \models b$ means that boolean expression b holds under the truth assignment represented by ℓ . The semantics of boolean expressions are defined as follows, where p denotes an atomic proposition and b , b_1 , and b_2 denote boolean expressions.

- $\ell \models p \iff p \in \ell$
- $\ell \models (b) \iff \ell \models b$
- $\ell \models \neg b \iff \ell \not\models b$
- $\ell \models b_1 \wedge b_2 \iff \ell \models b_1 \text{ and } \ell \models b_2$

2.1 Unlocked Semantics

Semantics of unlocked SEREs

The semantics of unlocked SEREs are defined over finite words from the alphabet 2^P . We will denote a finite word over 2^P by w . The concatenation of w_1 and w_2 is denoted

² Where $\top = p \vee \neg p$ for some $p \in P$.

by w_1w_2 . The empty word is denoted by ϵ , so that $w\epsilon = \epsilon w = w$. The notation $w \models r$, where r is a SERE, means that w is in the language of r . The semantics of SEREs are defined as follows, where b denotes a boolean expression, r , r_1 , and r_2 denote unlocked SEREs, and $[i..k]$ denotes the set of integers $\{j : i \leq j \wedge j \leq k\}$.

- $w \models b \iff |w| = 1$ and $\ell_0 \models b$
- $w \models \{r\} \iff w \models r$
- $w \models r_1; r_2 \iff$ there exist w_1 and w_2 such that $w = w_1w_2$, $w_1 \models r_1$, and $w_2 \models r_2$
- $w \models r_1:r_2 \iff$ there exist w_1 , w_2 , and ℓ such that $w = w_1\ell w_2$, $w_1\ell \models r_1$, and $\ell w_2 \models r_2$
- $w \models \{r_1\}|\{r_2\} \iff w \models r_1$ or $w \models r_2$
- $w \models \{r_1\} \&\& \{r_2\} \iff w \models r_1$ and $w \models r_2$
- $w \models r[*] \iff$ either $w = \epsilon$ or there exist w_1, w_2, \dots, w_j such that $w = w_1w_2 \dots w_j$ and for every $i \in [1..j]$, $w_i \models r$

Semantics of unlocked Sugar FL formulas

The semantics of Sugar FL formulas are defined over finite or infinite words from the alphabet 2^P . The notation $\omega \models f$ means that formula f holds along the (finite or infinite) word ω . The notation $M \models f$ means that $\hat{L}(\pi) \models f$ for every computation path π in M such that $\pi_0 \in S_0$. The semantics of an FL formula are defined as follows³, where b denotes a boolean expression, r , r_1 , and r_2 denote SEREs, f , f_1 , and f_2 denote FL formulas, and $[i..k)$ denotes the set of integers $\{j : i \leq j \wedge j < k\}$.

- $\omega \models b \iff \ell_0 \models b$
- $\omega \models (f) \iff \omega \models f$
- $\omega \models \neg f \iff \omega \not\models f$
- $\omega \models f_1 \wedge f_2 \iff \omega \models f_1$ and $\omega \models f_2$
- $\omega \models X! f \iff |\omega| > 1$ and $\omega^1 \models f$
- $\omega \models [f_1 U f_2] \iff$ there exists $k \in [0..|\omega|)$ such that $\omega^k \models f_2$, and for every $j \in [0..k)$, $\omega^j \models f_1$
- $\omega \models \{r\}(f) \iff$ for every $j \in [0..|\omega|)$ such that $\omega^{0,j} \models r$, $\omega^j \models f$
- $\omega \models \{r_1\} \mapsto \{r_2\}! \iff$ for every $j \in [0..|\omega|)$ such that $\omega^{0,j} \models r_1$ there exists $k \in [j..|\omega|)$ such that $\omega^{j,k} \models r_2$
- $\omega \models \{r_1\} \mapsto \{r_2\} \iff$ for every $j \in [0..|\omega|)$ such that $\omega^{0,j} \models r_1$ either there exists $k \in [j..|\omega|)$ such that $\omega^{j,k} \models r_2$ or for every $k \in [j..|\omega|)$ there exists a finite word ω' such that $\omega^{j,k}\omega' \models r_2$
- $\omega \models f$ abort $b \iff$ either $\omega \models f$ or $\omega \models b$ or there exists $j \in [1..|\omega|)$ and word ω' such that $\omega^j \models b$ and $\omega^{0,j-1}\omega' \models f$

³ The semantics presented here for the LTL operators are the standard ones.

2.2 Clocked Semantics

In the above we disregarded the *clock operator* ($@$) in the definition of Sugar formulas (and SEREs). The semantics of clocked SEREs and clocked Sugar formulas are defined formally below⁴.

Semantics of clocked SEREs

Clocked SEREs are defined over finite words from the alphabet 2^P and a boolean expression that serves as the clock context. The notation $w \models^c r$, where r is a SERE and c is a boolean expression, means that w is in the language of r in context of clock c . The semantics of clocked SEREs are defined as follows, where b , c , and c_1 denote boolean expressions, r , r_1 , and r_2 denote clocked SEREs, and $[i..k]$ denotes the set of integers $\{j : i \leq j \wedge j < k\}$.

- $w \models^c b \iff |w| \geq 1$, for every $i \in [0..|w| - 1]$, $\ell_i \models \neg c$ and $\ell_{|w|-1} \models c \wedge b$
- $w \models^c \{r\} \iff w \models^c r$
- $w \models^c r_1; r_2 \iff$ there exists w_1 and w_2 such that $w = w_1 w_2$, $w_1 \models^c r_1$, and $w_2 \models^c r_2$
- $w \models^c r_1 : r_2 \iff$ there exists w_1 , w_2 , and ℓ such that $w = w_1 \ell w_2$, $w_1 \ell \models^c r_1$, and $\ell w_2 \models^c r_2$
- $w \models^c \{r_1\} | \{r_2\} \iff w \models^c r_1$ or $w \models^c r_2$
- $w \models^c \{r_1\} \&\& \{r_2\} \iff w \models^c r_1$ and $w \models^c r_2$
- $w \models^c r[*] \iff$ either $w = \epsilon$ or there exists w_1, w_2, \dots, w_j such that $w = w_1 w_2 \dots w_j$ and for every $i \in [1..j]$, $w_i \models^c r$
- $w \models^c r@c_1 \iff$ there exists $i \in [0..|w|)$ such that $\omega^{0,i} \models \{\neg c_1[*]; c_1\}$ and $\omega^i \models^c r$

Semantics of clocked Sugar FL formulas

We now turn to the semantics of clocked Sugar FL formulas. The notation $\omega \models^c f$ where f is a formula and c is a boolean expression means that formula f holds along the (finite or infinite) word ω in the context of clock c . The notation $M \models f$ means that $\hat{L}(\pi) \models^T f$ for every computation path π in M such that $\pi_0 \in S_0$ (where $T = p \vee \neg p$ for some $p \in P$). The semantics of a (clocked) Sugar FL formula are defined as follows⁵, where b , c , and c_1 denote boolean expressions, r , r_1 , and r_2 denote SEREs, f , f_1 , and f_2 denote (clocked) FL formulas, $[i..k]$ denotes the set of integers $\{j : i \leq j \wedge j < k\}$, and $(i..k)$ denotes the set of integers $\{j : i < j \wedge j < k\}$.

- $\omega \models^c b \iff \omega \models b$
- $\omega \models^c (f) \iff \omega \models^c f$

⁴ An equivalent definition in terms of rewrite rules is given in Appendix A.

⁵ When the context is \top , the semantics reduce to the unclocked semantics as previously presented. Thus, the semantics of the LTL operators in context \top reduce to the standard ones.

- $\omega \models^c \neg f \iff \omega \not\models^c f$
- $\omega \models^c f_1 \wedge f_2 \iff \omega \models^c f_1$ and $\omega \models^c f_2$
- $\omega \models^c X! f \iff$ there exists $i \in [1..|\omega|)$ such that $\omega^{1,i} \models^T \{\neg c[*]; c\}$ and $\omega^i \models^c f$
- $\omega \models^c [f_1 U f_2] \iff$ there exists $k \in [0..|\omega|)$ such that $\omega^k \models^T c$, $\omega^k \models^c f_2$, and for every $j \in [0..k)$ for which $\omega^j \models^T c$, $\omega^j \models^c f_1$
- $\omega \models^c \{r\}(f) \iff$ for every $i \in [0..|\omega|)$ such that $\omega^{0,i} \models^c r$, there exists $j \in [i..|\omega|)$ such that $\omega^j \models^c f$
- $\omega \models^c \{r_1\} \mapsto \{r_2\}! \iff$ for every $i \in [0..|\omega|)$ such that $\omega^{0,i} \models^c r_1$ there exists $j \in [i..|\omega|)$ such that $\omega^{i,j} \models^c r_2$
- $\omega \models^c \{r_1\} \mapsto \{r_2\} \iff$ for every $i \in [0..|\omega|)$ such that $\omega^{0,i} \models^c r_1$, either there exists $j \in [i..|\omega|)$ such that $\omega^{i,j} \models^c r_2$ or for every $j \in [i..|\omega|)$ there exists a finite word ω' such that $\omega^{i,j}\omega' \models^c r_2$
- $\omega \models^c f$ abort $b \iff$ either $\omega \models^c f$ or $\omega \models^c b$ or there exists $i \in [1..|\omega|)$ and word ω' such that $\omega^i \models c \wedge b$ and $\omega^{0,i-1}\omega' \models^c f$
- $\omega \models^c f@c_1! \iff$ there exists $i \in [0..|\omega|)$ such that $\omega^{0,i} \models \{\neg c_1[*]; c_1\}$ and $\omega^i \models^{c_1} f$

2.3 Semantics of OBE formulas

The semantics of OBE formulas are defined over states in the model, rather than finite or infinite words. The notation $M, s \models f$ means that formula f holds in state s of model M . The notation $M \models f$ is equivalent to $\forall s \in S_0 : M, s \models f$. In other words, f is valid for every initial state of M . The semantics of an OBE formula are defined as follows⁶, where b denotes a boolean expression and f , f_1 , and f_2 denote OBE formulas.

- $M, s \models b \iff s \models b$
- $M, s \models (f) \iff M, s \models f$
- $M, s \models \neg f \iff M, s \not\models f$
- $M, s \models f_1 \wedge f_2 \iff M, s \models f_1$ and $M, s \models f_2$
- $M, s \models EX f \iff$ there exists a computation path π of M such that $|\pi| > 1$, $\pi_0 = s$, and $M, \pi_1 \models f$
- $M, s \models E[f_1 U f_2] \iff$ there exists a computation path π of M such that $\pi_0 = s$ and there exists $k < |\pi|$ such that $M, \pi_k \models f_2$ and for every j such that $j < k$: $M, \pi_j \models f_1$
- $M, s \models EG f \iff$ there exists a computation path π of M such that $\pi_0 = s$ and for every j such that $0 \leq j < |\pi|$: $M, \pi_j \models f$

3 Syntactic sugaring

The remainder of the temporal layer is syntactic sugar. In other words, it does not add expressive power, and every piece of syntactic sugar can be defined in terms of the basic Sugar FL operators presented above. The syntactic sugar is defined below⁷.

⁶ The semantics are those of standard CTL.

⁷ Where $\text{T} = p \vee \neg p$ for some $p \in P$ and $\text{F} = p \wedge \neg p$ for some $p \in P$.

Note: the definitions given here do not necessarily represent the most efficient implementation. In some cases, there is an equivalent syntactic sugaring, or a direct implementation, that is more efficient.

Additional SERE operators

If i, j, k , and l are integer constants such that $i \geq 0, j \geq i, k \geq 1$ and $l \geq k$, then additional SERE operators can be viewed as abbreviations of the basic SERE operators defined above, as follows, where b denotes a boolean expression, and r denotes a SERE.

- $\{r_1\} \& \{r_2\} = \{\{r_1\} \&\& \{r_2; T[*]\}\} \mid \{\{r_1; T[*]\} \&\& \{r_2\}\}$
- $r[+] = r; r[*]$
- $r[*i] = \begin{cases} F[*] & \text{if } i = 0 \\ \underbrace{r; r; \dots; r}_{i \text{ times}} & \text{otherwise} \end{cases}$
- $r[*i..j] = \{r[*i]\} \mid \dots \mid \{r[*j]\}$
- $r[*i..] = \{r[*i]\}; \{r[*]\}$
- $r[*..i] = \{r[*0]\} \mid \dots \mid \{r[*i]\}$
- $r[*..] = r[*0..]$
- $[+] = T[+]$
- $[*] = T[*]$
- $[*i] = T[*i]$
- $[*i..j] = T[*i..j]$
- $[*i..] = T[*i..]$
- $[*..i] = T[*..i]$
- $[*..] = T[*..]$
- $b[= i] = \{\neg b[*]; b[*i]; \neg b[*]\}$
- $b[= i..j] = \{b[= i]\} \mid \dots \mid \{b[= j]\}$
- $b[= i..] = b[= i]; [*]$
- $b[= ..i] = \{b[= 0]\} \mid \dots \mid \{b[= i]\}$
- $b[= ..] = b[= 0..]$
- $b[\rightarrow] = \neg b[*]; b$
- $b[\rightarrow k] = \{\neg b[*]; b[*k]\}$
- $b[\rightarrow k..l] = \{b[\rightarrow k]\} \mid \dots \mid \{b[\rightarrow l]\}$
- $b[\rightarrow k..] = \{b[\rightarrow k]\} \mid \{b[\rightarrow k]; [*]; b\}$
- $b[\rightarrow ..k] = \{b[\rightarrow 1]\} \mid \dots \mid \{b[\rightarrow k]\}$
- $b[\rightarrow ..] = b[\rightarrow 1..]$

Additional operators

If i, j, k and l are integers such that $i \geq 0, j \geq i, k > 0$ and $l \geq k$ then additional operators can be viewed as abbreviations of the basic operators defined above, as follows, where b denotes a boolean expression, r, r_1 , and r_2 denote SEREs, and f, f_1 , and f_2 denote FL formulas.

- $f_1 \vee f_2 = \neg(\neg f_1 \wedge \neg f_2)$

- $f_1 \rightarrow f_2 = \neg f_1 \vee f_2$
- $f_1 \leftrightarrow f_2 = (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$

- $Gf = \neg F\neg f$
- $Xf = \neg X! \neg f$
- $Ff = [\top U f]$
- $[f_1 W f_2] = [f_1 U f_2] \vee Gf_1$

- *always* $f = G f$
- *never* $f = G \neg f$
- *next!* $f = X! f$
- *next* $f = X f$
- *eventually!* $f = Ff$
- f_1 *releases* $f_2 = [f_1 V f_2]$

- f_1 *until!* $f_2 = [f_1 U f_2]$
- f_1 *until* $f_2 = [f_1 W f_2]$
- f_1 *until!*₋ $f_2 = [f_1 U f_1 \wedge f_2]$
- f_1 *until*₋ $f_2 = [f_1 W f_1 \wedge f_2]$

- f_1 *before!* $f_2 = [\neg f_2 U f_1 \wedge \neg f_2]$
- f_1 *before* $f_2 = [\neg f_2 W f_1 \wedge \neg f_2]$
- f_1 *before!*₋ $f_2 = [\neg f_2 U f_1]$
- f_1 *before*₋ $f_2 = [\neg f_2 W f_1]$

- $X! [i]f = \overbrace{X! X! \dots X!}^{i \text{ times}} f$
- $X[i]f = \overbrace{XX\dots X}^{i \text{ times}} f$
- *next!*_[i] $f = X! [i] f$
- *next*_[i] $f = X [i] f$
- *next_a!*_[i..j] $f = (X! [i]f) \wedge \dots \wedge (X! [j]f)$
- *next_a*_[i..j] $f = (X [i]f) \wedge \dots \wedge (X [j]f)$
- *next_e!*_[i..j] $f = (X! [i]f) \vee \dots \vee (X! [j]f)$
- *next_e*_[i..j] $f = (X [i]f) \vee \dots \vee (X [j]f)$

- *next_event!*_(b) $f = [\neg b U b \wedge f]$
- *next_event*_(b) $f = [\neg b W b \wedge f]$

- *next_event!*_(b)_[k] $f = \text{next_event!}(b) \overbrace{(X! \text{next_event!}(b) \dots (X! \text{next_event!}(b)(f)) \dots)}^{k-1 \text{ times}}$
- *next_event*_(b)_[k] $f = \text{next_event}(b) \overbrace{(X \text{next_event}(b) \dots (X \text{next_event}(b)(f)) \dots)}^{k-1 \text{ times}}$
- *next_event_a!*_(b)_[k..l] $f = \text{next_event!}(b)[k](f) \wedge \dots \wedge \text{next_event!}(b)[l](f)$
- *next_event_a*_(b)_[k..l] $f = \text{next_event}(b)[k](f) \wedge \dots \wedge \text{next_event}(b)[l](f)$
- *next_event_e!*_(b)_[k..l] $f = \text{next_event!}(b)[k](f) \vee \dots \vee \text{next_event!}(b)[l](f)$

- $next_event_e(b)[k..l](f) = next_event(b)[k](f) \vee \dots \vee next_event(b)[l](f)$
- $\{r_1\} \Rightarrow \{r_2\}! = \{r_1\} \mapsto \{\mathbb{T}; r_2\}!$
- $\{r_1\} \Rightarrow \{r_2\} = \{r_1\} \mapsto \{\mathbb{T}; r_2\}$
- $always\{r\} = \{\mathbb{T}[*]\} \mapsto \{r\}$
- $never\{r\} = \{\mathbb{T}[*]; r\} \mapsto \{\mathbb{F}\}$
- $eventually\{r\} = \{\mathbb{T}\} \mapsto \{\mathbb{T}[*]; r\}!$
- $within!(r_1, b)\{r_2\} = \{r_1\} \mapsto \{r_2 \ \&\& \ b[= 0]; b\}!$
- $within(r_1, b)\{r_2\} = \{r_1\} \mapsto \{r_2 \ \&\& \ b[= 0]; b\}$
- $within!(r_1, b)\{r_2\} = \{r_1\} \mapsto \{r_2 \ \&\& \ \{b[= 0]; b\}\}!$
- $within_(r_1, b)\{r_2\} = \{r_1\} \mapsto \{r_2 \ \&\& \ \{b[= 0]; b\}\}$
- $whilenot!(b)\{r\} = within!(\mathbb{T}, b)\{r\}$
- $whilenot(b)\{r\} = within(\mathbb{T}, b)\{r\}$
- $whilenot!(b)\{r\} = within!(\mathbb{T}, b)\{r\}$
- $whilenot_(b)\{r\} = within_(\mathbb{T}, b)\{r\}$
- $f@c = \neg(\neg f@c!)$

Forall

If f is a Sugar formula, v_0, v_1, \dots, v_n are constants, and j, k, l and m are integers, then the following are Sugar formulas:

- $forall\ i\ in\ \{v_0, v_1, \dots, v_n\} : f$
- $forall\ i\ in\ j..k : f$
- $forall\ i\ in\ boolean : f$
- $forall\ i\langle l..m \rangle\ in\ \{v_0, v_1, \dots, v_n\} : f$
- $forall\ i\langle l..m \rangle\ in\ j..k : f$
- $forall\ i\langle l..m \rangle\ in\ boolean : f$

Forall does not add expressive power. Rather, it can be viewed as additional syntactic sugar, as follows:

- $forall\ i\ in\ \{v_0, v_1, \dots, v_n\} : f = \bigwedge_{u \in \{v_0, v_1, \dots, v_n\}} f[i \leftarrow u]$
- $forall\ i\ in\ j..k : f = \bigwedge_{u=j}^k f[i \leftarrow u]$
- $forall\ i\ in\ boolean : f = \bigwedge_{u=0}^1 f[i \leftarrow u]$
- $forall\ i\langle l..m \rangle\ in\ \{v_0, v_1, \dots, v_n\} : f = \bigwedge_{u_l \in \{v_0, v_1, \dots, v_n\}} \dots \bigwedge_{u_m \in \{v_0, v_1, \dots, v_n\}} f[i\langle l..m \rangle \leftarrow \langle u_l..u_m \rangle]$
- $forall\ i\langle l..m \rangle\ in\ j..k : f = \bigwedge_{u_l=j}^k \dots \bigwedge_{u_m=j}^k f[i\langle l..m \rangle \leftarrow \langle u_l..u_m \rangle]$

$$- \text{forall } i\langle l..m \rangle \text{ in boolean : } f = \bigwedge_{u_l=0}^1 \dots \bigwedge_{u_m=0}^1 f[i\langle l..m \rangle \leftarrow \langle u_l..u_m \rangle]$$

where $f[i \leftarrow u]$ is the formula obtained from f by replacing every occurrence of i by u and $f[i\langle l..m \rangle \leftarrow \langle u_l..u_m \rangle]$ is the formula obtained from f by replacing every occurrence of index j (where $l \leq j \leq m$) in the vector i by u_j .

4 Typed-text representation of symbols used in this definition

Table 1 shows the mapping of various symbols used in this definition to the corresponding typed-text Sugar representation.

Table 1. Typed-text symbols in the Verilog, VHDL, and EDL flavors

	Verilog	VHDL	EDL
\mapsto	->	->	->
\mapsto	=>	=>	=>
\rightarrow	->	->	->
\leftrightarrow	<->	<->	<->
\neg	!	not	!
\wedge	&&	and	&
\vee		or	
$..$:	to	..
$\langle \rangle$	[]	()	()

Acknowledgements

Thank you to Sharon Barner, Shoham Ben-David, Alan Hartman, and Emmanuel Zarpas for their help in the formal definition of multiple clocks.

A Rewriting rules for clocks

In Section 2.2 we gave the semantics of clocked Sugar formulas directly. There is an equivalent definition in terms of unclocked Sugar formulas, as follows: Starting from the outermost clock, use the following rules to translate clocked SEREs into unclocked SEREs, and clocked Sugar formulas into unclocked Sugar formulas. The rewrite rules for SEREs are:

1. $\mathcal{T}^c(b) = \{-c[*]; c \wedge b\}$

2. $\mathcal{T}^c(r_1 ; r_2) = \mathcal{T}^c(r_1) ; \mathcal{T}^c(r_2)$
3. $\mathcal{T}^c(r_1 : r_2) = \mathcal{T}^c(r_1) : \mathcal{T}^c(r_2)$
4. $\mathcal{T}^c(r_1 \mid r_2) = \mathcal{T}^c(r_1) \mid \mathcal{T}^c(r_2)$
5. $\mathcal{T}^c(r_1 \&\& r_2) = \mathcal{T}^c(r_1) \&\& \mathcal{T}^c(r_2)$
6. $\mathcal{T}^c(r[*]) = \{\mathcal{T}^c(r)\}[*]$
7. $\mathcal{T}^c(r@c_1) = \{\neg c_1[*]; c_1:\mathcal{T}^{c_1}(r_1)\}$

The rewriting rules for Sugar formulas are:

1. $\mathcal{T}^c(b) = b$
2. $\mathcal{T}^c(\neg f) = \neg \mathcal{T}^c(f)$
3. $\mathcal{T}^c(f_1 \wedge f_2) = (\mathcal{T}^c(f_1) \wedge \mathcal{T}^c(f_2))$
4. $\mathcal{T}^c(\mathbf{X}!f) = \mathbf{X}! [\neg c \mathbf{U} (c \wedge \mathcal{T}^c(f))]$
5. $\mathcal{T}^c(f_1 \mathbf{U} f_2) = [(c \rightarrow \mathcal{T}^c(f_1)) \mathbf{U} (c \wedge \mathcal{T}^c(f_2))]]$
6. $\mathcal{T}^c(\{r\}(f)) = \{\mathcal{T}^c(r)\}([\neg c \mathbf{U} (c \wedge \mathcal{T}^c(f))])$
7. $\mathcal{T}^c(\{r_1\} \mapsto \{r_2\}!) = \{\mathcal{T}^c(r_1)\} \mapsto \{\mathcal{T}^c(r_2)\}!$
8. $\mathcal{T}^c(\{r_1\} \mapsto \{r_2\}) = \{\mathcal{T}^c(r_1)\} \mapsto \{\mathcal{T}^c(r_2)\}$
9. $\mathcal{T}^c(f \text{ abort } b) = \mathcal{T}^c(f) \text{ abort } (c \wedge b)$
10. $\mathcal{T}^c(f@c_1!) = [\neg c_1 \mathbf{U} (c_1 \wedge \mathcal{T}^{c_1}(f))]$