H-0256 (H0801-006) January 6, 2008
Computer Science

IBM Research Report

Proposal for Extending Annex B of PSL with Local Variables, Procedural Blocks, Past Expressions and Clock Alignment Operators

Cindy Eisner, Dana Fisman*
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

*also with Hebrew University

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Proposal for Extending Annex B of PSL with Local Variables, Procedural Blocks, Past Expressions and Clock Alignment Operators

Cindy Eisner[1] and Dana Fisman[1,2]

[1] IBM Haifa Research Lab
[2] Hebrew University

December 3, 2007

Abstract

In this document we extend Proposal A' for augmenting the formal semantics of PSL with local variables to all of PSL (issue 67). Proposal A' provides semantics for PSL without truncated words, abort and clock operators. In this proposal we augment the entire formal semantics of PSL (including the above constructs) with local variables. In addition, we add semantics for procedural blocks (issue 66), for built-in functions that refer to the past (issue 117) and for clock alignment operators (issue 137).

This document presents the semantics gradually. First it presents the semantics of the *base logic*, which is the set of FL formulas that (i) do not use the clock operator (ii) do not refer to the past (i.e., do not use past built-in functions) and (iii) do not use local variables or procedural blocks. Then it presents the semantics of the base logic extended with either the clock operator or references to the past or local variables and procedural blocks. Last it presents the overall semantics of linear formulas with no restrictions.

Annex B

(normative)

Formal Syntax and Semantics of IEEE Std 1850 Property Specification Language (PSL)

This appendix formally describes the syntax and semantics of the temporal layer.

B.1 Typed-text representation of symbols

Table 1 shows the mapping of various symbols used in this definition to the corresponding typed-text PSL representation, in the different flavors.

NOTE –
For reasons of simplicity, the syntax given herein is more flexible than the one defined by the extended BNF (given in Annex A). That is, some of the expressions which are legal here are not legal under the BNF Grammar. Users should use the stricter syntax, as defined by the BNF grammar in Annex A.

Table 1: Typed-text symbols in the SystemVerilog, Verilog, VHDL, SystemC and GDL flavors

	SystemVerilog	Verilog	VHDL	SystemC	GDL
$\mapsto$	`|->`	`|->`	`|->`	`|->`	`|->`
$\Rightarrow$	`|=>`	`|=>`	`|=>`	`|=>`	`|=>`
$\rightarrow$	`->`	`->`	`->`	`->`	`->`
$\leftrightarrow$	`<->`	`<->`	`<->`	`<->`	`<->`
$\neg$	`!`	`!`	`not`	`!`	`!`
$\wedge$	`&&`	`&&`	`and`	`&&`	`&`
$\vee$	`||`	`||`	`or`	`||`	`|`
..	`:`	`:`	`to`	`:`	`..`
$\langle\,\rangle$	`[ ]`	`[ ]`	`( )`	`( )`	`( )`

B.2 Syntax

The logic PSL is defined with respect to a non-empty finite set of atomic propositions AP, a finite set of local variables LV with domain D, a given set of (unary and binary) operators OP over the domain D and a given set of procedural blocks PB with local variables in LV as input parmeters. We assume that T and F belong to D.

Definition 1 (Expressions and Boolean expressions)
The set E of expressions is defined as follows:

1. *Every atomic proposition $p \in AP$ is an expression in E.*
2. *Every local variable $v \in LV$ is an expression in E.*
3. *If e is an expression and $\circ$ a unary operator in OP then $\circ e$ is an expression in E.*
4. *If e_1, e_2 are expressions and $\otimes$ a binary operator in OP then $e_1 \otimes e_2$ is an expression in E.*
5. *If e is an expression, c a Boolean expression and n a non-negative integer then $prev(e)$, $prev(e, n)$ and $prev(e, n, c)$ are expressions in E.*
6. *If r is a* SERE *and c a Boolean expression then $ended(r)$ and $ended(r, c)$ are (Boolean) expressions in E.*

We refer to an expression whose domain is $\{\mathrm{T}, \mathrm{F}\}$ as a Boolean expression and use B to denote the set of Boolean expressions. We use $\mathsf{B}_{-\mathrm{LV}}$ to denote the set of Boolean expressions that do not use local variables. We use $\mathsf{B}_{-\mathrm{PE}}$ to denote the set of Boolean expressions that do not use past expressions (that is, $prev(\cdot)$, $ended(\cdot)$ and their derivatives as per Section B.4.0). We use $\mathsf{B}_{-\mathrm{LV,PE}}$ to denote the set of Boolean expressions that do not use local variables or past expressions. We often refer to Boolean expressions in $\mathsf{B}_{-\mathrm{LV,PE}}$ as basic *Boolean expressions. Note that we have $\mathsf{B} \subseteq \mathsf{E}$, $\mathsf{B}_{-\mathrm{LV,PE}} \subseteq \mathsf{B}_{-\mathrm{LV}} \subseteq \mathsf{B}$ and $\mathsf{B}_{-\mathrm{LV,PE}} \subseteq \mathsf{B}_{-\mathrm{PE}} \subseteq \mathsf{B}$.*

Definition 2 (Sequential Extended Regular Expressions (SEREs))

1. *Every Boolean expression $b \in \mathsf{B}$ is a* SERE.
2. *If b is a Boolean expression in B, $e_1, \ldots, e_n$ are expressions in E, and $y_1, \ldots, y_n$ are local variables in LV then the following is a* SERE*:*

$$b, y_1 \leftarrow e_1, \ldots, y_n \leftarrow e_n$$

3. *If r, r_1, and r_2 are* SEREs*, c is a Boolean expression in $\mathsf{B}_{-\mathrm{LV}}$, and $z \in LV$ is a local variable then the following are* SEREs*:*
   - $\{r\}$ • $r_1 \; ; \; r_2$ • $r_1 : r_2$ • $r_1 \mid r_2$ • $r_1 \;\&\&\; r_2$
   - $[*0]$ • $r[*]$ • $r\,@c$ • $\{\mathsf{var}(z)\ r\}$ • $\{\mathsf{free}(z)\ r\}$

Definition 3 (Formulas of the Foundation Language (FL formulas))
If φ and ψ are FL formulas, n a non-negative integer, r is a SERE*, $a, c \in \mathsf{B}_{-\mathrm{LV}}$ Boolean expressions over AP, and $z \in LV$ is a local variable then the following are FL formulas:*

- (φ) • $\neg\varphi$ • $\varphi \wedge \psi$ • $r!$ • r • $r \mapsto \varphi$
- $\mathsf{X}![n]\ \varphi$ • $\varphi\ \mathsf{U}\ \psi$ • $\varphi @c$ • $\varphi\ \mathsf{async\_abort}\ a$ • $\varphi\ \mathsf{sync\_abort}\ a$ • $(\mathsf{var}(z)\ \varphi)$

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

1. *Every basic Boolean expression $b \in \mathsf{B}_{-\mathrm{LV,PE}}$ is an OBE formula.*
2. *If f, f_1, and f_2 are OBE formulas, then so are the following:*
   (a) (f)
   (b) $\neg f$
   (c) $f_1 \wedge f_2$
   (d) $\mathsf{EX} f$
   (e) $\mathsf{E}[f_1\ \mathsf{U}\ f_2]$
   (f) $\mathsf{EG} f$

Definition 5 (PSL Formulas)

1. *Every FL formula is a PSL formula.*
2. *Every OBE formula is a PSL formula.*

In Section B.4, we show additional operators which provide syntactic sugaring to the ones above.

B.3 Semantics

The semantics of PSL formulas are defined with respect to a *model*. A model is a quintuple (S, S_0, R, AP, 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, AP is a non-empty set of atomic propositions, and L is the valuation, a function $L : S \longrightarrow 2^{AP}$, mapping each state with a set of atomic propositions valid in that state.

A *path* π is a possibly empty finite (or infinite) sequence of states $\pi = (\pi_0, \pi_1, \pi_2, \cdots, \pi_n)$ (or $\pi = (\pi_0, \pi_1, \pi_2, \cdots)$). A *computation path* π of a model M is a non-empty finite (or infinite) path π such that $\pi_0 \in S_0$ and 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 overload L, to denote the extension of the valuation function L from states to paths as follows: $L(\pi) = L(\pi_0)L(\pi_1)\ldots L(\pi_n)$ (or $L(\pi) = L(\pi_0)L(\pi_1)\ldots$). Thus we have a mapping from *states* in M to *letters* of 2^{AP}, and from finite (or infinite) *paths* in M to finite (or infinite) *words* over 2^{AP}.

Let v be an empty/finite/infinite word over some alphabet. We denote the length of word v as $|v|$. A finite non-empty word $v = (\ell_0\ell_1\ell_2\cdots\ell_n)$ has length $n+1$, the (finite) empty word $v = \epsilon$ has length 0, and an infinite word has length ∞. We use i, j, and k to denote non-negative integers. We denote the i^{th} letter of v by v^{i-1} (since counting of letters starts at zero). We denote by $v^{i\cdot\cdot}$ the suffix of v starting at v^i. That is, for every $i < |v|$, $v^{i\cdot\cdot} = v^i v^{i+1}\cdots v^n$ or $v^{i\cdot\cdot} = v^i v^{i+1}\cdots$. We denote by $v^{i..j}$ the finite sequence of letters starting from v^i and ending in v^j. That is, for $j \geq i$, $v^{i..j} = v^i v^{i+1}\cdots v^j$ and for $j < i$, $v^{i..j} = \epsilon$. We use ℓ^ω to denote an infinite-length word, each letter of which is ℓ.

B.3.1 Semantics of FL formulas

This document presents the semantics of FL formulas gradually. First it presents the semantics of the *base logic*, which is the set of FL formulas that (i) do not use the clock operator (ii) do not refer to the past (i.e., do not use $prev(\cdot)$ or $ended(\cdot)$) and (iii) do not use local variables or procedural blocks. Then it presents the semantics of the base logic extended with either the clock operator or references to the past or local variables and procedural blocks. Last it presents the overall semantics of FL formulas.

B.3.1.1 Semantics of the Base Logic

In this section we provide semantics of FL formulas that (i) do not use the clock operator (ii) do not refer to the past and (iii) do not use local variables or procedural blocks, which we refer to as the *base semantics*.

Let AP be a set of atomic propositions and let Σ be the set of all possible assignments of the atomic propositions AP (i.e. $\Sigma = 2^{AP}$). Let $\widehat{\Sigma}$ denote the alphabet $\Sigma \cup \{\top, \bot\}$.

In the sequel we use u, v, w to denote words over $\widehat{\Sigma}$ and σ to denote a letter in $\widehat{\Sigma}$.

Given a letter σ in $\widehat{\Sigma}$ we use $\overline{\sigma}$ to denote the *dual* of σ which is $\bot$ if σ is $\top$, is $\top$ if σ is $\bot$ and is σ otherwise. We use $\overline{v}$ to denote the obvious extension of the definition of dual to words.

B.3.1.1.0 Base Semantics of Expressions

The semantics of FL *formulas* is defined inductively, using as the base case the semantics of *expressions*. In the case of the base semantics, we neglect local variables, and the built-in functions $ended(\cdot)$ and $prev(\cdot)$. Thus we have only basic Boolean expressions ($\mathsf{B}_{\text{–LV,PE}}$) to consider. We view a basic Boolean expression $b \in \mathsf{B}_{\text{–LV,PE}}$ as a mapping $b : \widehat{\Sigma} \mapsto \{\text{T}, \text{F}\}$. Thus, $b(\sigma)$ denotes the value of b under $\sigma \in \widehat{\Sigma}$.

Let $\sigma \in \Sigma$. When b is an atomic proposition, say b is p for some $p \in AP$, then $p(\sigma)$ is equivalent to $p \in \sigma$. We define true and false to be the basic Boolean expressions that map every $\sigma \in \Sigma$ to T and F, respectively. We assume that operators are closed under $\{\text{T}, \text{F}\}$ and that they behave in the usual manner over letters in the alphabet Σ, i.e. that for $\sigma \in \Sigma$, $b, b_1, b_2 \in \mathsf{B}_{\text{–LV,PE}}$, a binary operator $\otimes$ and a unary operator $\circ$ we have $b_1(\sigma) \otimes b_2(\sigma) = (b_1 \otimes b_2)(\sigma)$ and $\circ(b(\sigma)) = (\circ b)(\sigma)$. In particular we assume that Boolean disjunction, conjunction and negation behave in the usual manner.

Regarding the letters $\top$ and $\bot$, we assume that any basic Boolean expression b is evaluated to T on ($\top$) and to F on ($\bot$). That is $b(\top) = \text{T}$ and $b(\bot) = \text{F}$ for any basic Boolean expression b. Note that in particular we have $\mathsf{true}(\top) = \mathsf{false}(\top) = \text{T}$ and $\mathsf{true}(\bot) = \mathsf{false}(\bot) = \text{F}$.

B.3.1.1.1 Base Semantics of SEREs

The notation $v \models r$, where r is a base SERE and v a finite word means that v *models tightly* r. The semantics of base SEREs are defined as follows, where b denotes a basic Boolean expression in $\mathsf{B}_{\text{–LV,PE}}$, and r, r_1, and r_2 denote base SEREs:

1. $v \models \{r\} \iff v \models r$
2. $v \models b \iff |v| = 1$ and $b(v^0)$=T
3. $v \models r_1 \ ; \ r_2 \iff \exists v_1, v_2$ s.t. $v = v_1 v_2$, $v_1 \models r_1$, and $v_2 \models r_2$
4. $v \models r_1 : r_2 \iff \exists v_1, v_2$, and σ s.t. $v = v_1 \sigma v_2$, $v_1 \sigma \models r_1$, and $\sigma v_2 \models r_2$
5. $v \models r_1 \mid r_2 \iff v \models r_1$ or $v \models r_2$
6. $v \models r_1 \ \ \&\& \ \ r_2 \iff v \models r_1$ and $v \models r_2$

7. $v \models\!\!\models [*0] \iff v = \epsilon$

8. $v \models\!\!\models r[*] \iff$ either $v \models\!\!\models [*0]$ or $\exists v_1, v_2$ s.t. $v_1 \neq \epsilon$, $v = v_1 v_2$, $v_1 \models\!\!\models r$ and $v_2 \models\!\!\models r[*]$

B.3.1.1.2 Base Semantics of FL Formulas

Let v be a possibly empty finite or infinite word over $\widehat{\Sigma}$, a be a basic Boolean expression in $\mathsf{B}_{-\mathrm{LV,PE}}$, r a base SERE, φ, ψ FL formulas, and n a non-negative integer. The notations $v \models \varphi$ means that v *models* (or *satisfies*) φ.

1. $v \models (\varphi) \iff v \models \varphi$

2. $v \models \neg\varphi \iff \overline{v} \not\models \varphi$

3. $v \models \varphi \wedge \psi \iff v \models \varphi$ and $v \models \psi$

4. $v \models r! \iff \exists j < |v|$ s.t. $v^{0..j} \models\!\!\models r$

5. $v \models r \iff \forall j < |v|$, $v^{0..j}\top^\omega \models r!$

6. $v \models r \mapsto \varphi \iff \forall j < |v|$ s.t. $\overline{v}^{0..j} \models\!\!\models r$, $v^{j..} \models \varphi$

7. $v \models \mathsf{X!}[n]\ \varphi \iff |v| > n$ and $v^{n..} \models \varphi$

8. $v \models \varphi\ \mathsf{U}\ \psi \iff \exists k < |v|$ s.t. $v^{k..} \models \psi$, and $\forall j < k,\ \ v^{j..} \models \varphi$

9. $v \models \varphi$ $\mathsf{async\_abort}$ $a \iff$ either $v \models \varphi$ or $\exists j < |v|$ s.t. $a(v^j){=}\mathrm{T}$ and $v^{0..j-1}\top^\omega \models \varphi$

10. $v \models \varphi$ $\mathsf{sync\_abort}$ $a \iff v \models \varphi$ $\mathsf{async\_abort}$ a

NOTES –

1. The semantics given here for the LTL operators and the abort operators is equivalent to the truncated semantics given in [B1] which is interpreted over 2^P rather than over $2^P \cup \{\top, \bot\}$. Using $\models_\bullet$ for the semantics in [B1] the following proposition states the equivalence: Let w be a finite word over 2^P, and let φ be a formula of LTL^{trunc}. Then, as shown in [B5], the three following equivalences hold:
$$w \models^{-}_\bullet \varphi \iff w\top^\omega \models \varphi$$
$$w \models_\bullet \varphi \iff w \models \varphi$$
$$w \models^{+}_\bullet \varphi \iff w\bot^\omega \models \varphi$$

2. Using $\models_\bullet$ as in Note 1 above, we use *holds strongly* for $\models^{+}_\bullet$, *holds* for $\models_\bullet$, and *holds weakly* for $\models^{-}_\bullet$. The remaining terminology of Section 4.4.6 is formally defined as follows:
   - φ is *pending* on word w iff $w \models^{-}_\bullet \varphi$ and $w \not\models_\bullet \varphi$
   - φ *fails* on word w iff $w \not\models^{-}_\bullet \varphi$

3. There is a subtle difference between Boolean negation and formula negation. For instance, consider the formula $\neg b$. If $\neg$ is Boolean negation, then $\neg b$ holds on an empty path. If $\neg$ is formula negation, then $\neg b$ does not hold on an empty path. Rather than introduce distinct operators for Boolean and formula negation, we instead adopt the convention that negation applied to a Boolean expression is Boolean negation. This does not restrict expressivity, as formula negation of b can be expressed as $\{\neg b\}!$.

4. A justification of why the semantics of a weak SERE (e.g., r) is the appropriate definition given the semantics of a strong SERE (e.g. $r!$) can be found in [B6].

B.3.1.1.3 Base Semantics With Respect to a Model

Let φ be an FL formula and M a model. The notation $M \models \varphi$ means that for all π such that π is computation path of M, $L(\pi) \models \varphi$.

B.3.1.2 Semantics with Clocks

In this section we provide semantics of the base logic augmented with the clock operator, which we refer to as the *clocked semantics*. We use the same alphabet and notation as in the base semantics, and in addition we define a *clock tick* as follows.

Let c be a basic Boolean expression in $\mathsf{B}_{-\text{LV,PE}}$. We say that finite word v *is a clock tick of* c, denoted $tick_c(v)$, if $|v| > 0$ and $c(v^{|v|-1})$=T and for every non-negative integer $i < |v| - 1$, $c(v^i)$=F (where $c(\cdot)$ refers to the base semantics of expressions). For an integer $m > 0$ we say that finite word v *is* m *clock ticks of* c, denoted $ticks_c(v, m)$, if there exist m words $v_1, v_2, \ldots, v_m$ such that $v = v_1 v_2 \cdots v_m$ and for every $1 \leq i \leq m$ we have $tick_c(v_i)$.

B.3.1.2.0 Clocked Semantics of Expressions

The clocked semantics of expressions are the same as the base semantics of expressions.

B.3.1.2.1 Clocked Semantics of SEREs

The notation $v \models\!\!\models^c r$, where r is a SERE, c is a basic Boolean expression in $\mathsf{B}_{-\text{LV,PE}}$ and v a finite word, means that v models tightly r in the context of clock c. The semantics of clocked SEREs are defined as follows, where b, c, and c_1 denote basic Boolean expressions in $\mathsf{B}_{-\text{LV,PE}}$, and r, r_1, and r_2 denote clocked SEREs:

1. $v \models\!\!\models^c \{r\} \iff v \models\!\!\models^c r$
2. $v \models\!\!\models^c b \iff tick_c(v)$ and $b(v^{|v|-1})$=T
3. $v \models\!\!\models^c r_1 \ ; \ r_2 \iff \exists v_1, v_2$ s.t. $v = v_1 v_2$, $v_1 \models\!\!\models^c r_1$, and $v_2 \models\!\!\models^c r_2$
4. $v \models\!\!\models^c r_1 : r_2 \iff \exists v_1, v_2$, and σ s.t. $v = v_1 \sigma v_2$, $v_1 \sigma \models\!\!\models^c r_1$, and $\sigma v_2 \models\!\!\models^c r_2$
5. $v \models\!\!\models^c r_1 \mid r_2 \iff v \models\!\!\models^c r_1$ or $v \models\!\!\models^c r_2$
6. $v \models\!\!\models^c r_1 \ \ \&\& \ \ r_2 \iff v \models\!\!\models^c r_1$ and $v \models\!\!\models^c r_2$
7. $v \models\!\!\models^c [*0] \iff v = \epsilon$
8. $v \models\!\!\models^c r[*] \iff$ either $v \models\!\!\models^c [*0]$ or $\exists v_1, v_2$ s.t. $v_1 \neq \epsilon$, $v = v_1 v_2$, $v_1 \models\!\!\models^c r$ and $v_2 \models\!\!\models^c r[*]$
9. $v \models\!\!\models^c r @ c_1 \iff v \models\!\!\models^{c_1} r$

B.3.1.2.2 Clocked Semantics of FL Formulas

The semantics of clocked formulas is defined with respect to possibly empty finite/infinite words over $\widehat{\Sigma}$ and a basic Boolean expression c in $\mathsf{B}_{-\text{LV,PE}}$ which serves as the clock context. Let v be a possibly empty finite or infinite word, a, c, c_1 basic Boolean expressions in $\mathsf{B}_{-\text{LV,PE}}$, r a clocked SERE, φ, ψ clocked formulas, and n a non-negative integer. The notation $v \models^c \varphi$ means that v models φ in the context of clock c.

1. $v \models^c (\varphi) \iff v \models^c \varphi$

2. $v \models^c \neg\varphi \iff \overline{v} \not\models^c \varphi$

3. $v \models^c \varphi \wedge \psi \iff v \models^c \varphi$ and $v \models^c \psi$

4. $v \models^c r! \iff \exists j < |v|$ s.t. $v^{0..j} \models^c r$

5. $v \models^c r \iff \forall j < |v|$, $v^{0..j}\top^\omega \models^c r!$

6. $v \models^c r \mapsto \varphi \iff \forall j < |v|$ s.t. $\overline{v}^{0..j} \models^c r$, $v^{j..} \models^c \varphi$

7. $v \models^c \mathsf{X!}[n]\ \varphi \iff \exists j < |v|$ s.t. $ticks_c(v^{0..j}, n+1)$ and $v^{j..} \models^c \varphi$

8. $v \models^c \varphi\ \mathsf{U}\ \psi \iff \exists k < |v|$ s.t. $c(v^k)$=T, $v^{k..} \models^c \psi$, and $\forall j < k$ s.t. $c(v^j)$=T, $v^{j..} \models^c \varphi$

9. $v \models^c \varphi @ c_1 \iff v \models^{c_1} \varphi$

10. $v \models^c \varphi$ async_abort $a \iff$ either $v \models^c \varphi$ or $\exists j < |v|$ s.t. $a(v^j)$=T and $v^{0..j-1}\top^\omega \models^c \varphi$

11. $v \models^c \varphi$ sync_abort $a \iff$ either $v \models^c \varphi$ or $\exists j < |v|$ s.t. $(a \wedge c)(v^j)$=T and $v^{0..j-1}\top^\omega \models^c \varphi$

NOTES –

1. The clocked semantics for the LTL subset follows the clocks paper [B2], with the exception that strength is applied at the SERE level rather than at the propositional level.
2. The operator obtained by instantiating $\mathsf{X!}[n]$ with zero can be seen as an alignment operator, whose introduction is of both practical and theoretical importance [B7].

B.3.1.2.3 Clocked Semantics With Respect to a Model

Let φ be an FL formula and M a model. The notation $M \models \varphi$ means that for all π such that π is computation path of M, $L(\pi) \models^{\mathsf{true}} \varphi$.

B.3.1.3 Semantics with Past

In this section we provide semantics of the base logic augmented with past expressions, which we refer to as the *past augmented semantics*. We use the same alphabet and notation as in the base logic. We use the definition of clock ticks as in the clocked semantics.

B.3.1.3.0 Past Augmented Semantics of Expressions

In the case of the past augmented semantics, we do not have local variables. Thus we have only Boolean expressions over AP ($\mathsf{B}_{\text{-LV}}$) to consider.

The semantics of Boolean expressions over AP maps non-empty words over $\widehat{\Sigma}$ to the domain $\{\text{T}, \text{F}\}$. The last letter of the word represents the present, while the remainder of the word (without the last letter) represents the past. Let u and σ be a finite word and a letter over $\widehat{\Sigma}$, respectively. The semantics of Boolean expressions over AP (that may refer to the past) overloads the notation $b(\cdot)$ defined in the base semantics as follows (where $b(\sigma)$ below refers to the base semantics of expressions).

Let c be a basic Boolean expression in $\mathsf{B}_{\text{-LV}}$. We say that finite word v with past u *is a past clock tick of* c, denoted $past\_tick_c(u \cdot v)$, if $|v| > 0$ and $c(uv^0)$=T and for every non-negative integer $0 < i < |v|$, $c(uv^{0..i})$=F (where $c(\cdot)$ refers to the base semantics of expressions). For an integer $m > 0$ we say that finite word v with past u *is* m *past clock ticks of* c, denoted $past\_ticks_c(u \cdot v, m)$, if there exist m words $v_1, v_2, \ldots, v_m$ such that $v = v_1 v_2 \cdots v_m$ and for every $1 \leq i \leq m$ we have $past\_tick_c(uv_1 \ldots v_{i-1} \cdot v_i)$.

- If $b \in \mathsf{B}_{-\mathrm{LV,PE}}$ is a basic Boolean expression then $b(u\sigma) = b(\sigma)$.
- If r is a SERE and $c \in \mathsf{B}_{-\mathrm{LV}}$ is a Boolean expression then
  - $ended(r,c)(u\sigma) = \begin{cases} \top \text{ iff } \exists u_1, u_2 \text{ such that } u = u_1u_2 \text{ and } u_1 \cdot u_2\sigma \models r & \text{if } c{=}\mathsf{true} \\ ended(\mathcal{R}^c(r))(u\sigma) & \text{otherwise} \end{cases}$
  - $ended(r)(u\sigma) = ended(r, \mathsf{true})(u\sigma)$
- If $b, c \in \mathsf{B}_{-\mathrm{LV}}$ are Boolean expressions then
  - $prev(b,n,c)(u) = \begin{cases} e(u^{0..i}) & \text{if } \exists i < |u| \text{ such that } past\_ticks_c(u^{0..i-1} \cdot u^{i..|u|-1}, n+1) \\ \text{undefined} & \text{otherwise.} \end{cases}$
  - $prev(b,n)(u) = prev(b,n,\mathsf{true})(u)$
  - $prev(b)(u) = prev(b,1,\mathsf{true})(u)$

  where $\mathcal{R}^c(\cdot)$ is defined in Section B.5.

Note that $prev(e,n,c)(u)$ (as well as its derivatives as given in Section B.4.0) is undefined when u does not contain $n+1$ letters on which c holds.

B.3.1.3.1 Past Augmented Semantics of SEREs

The notation $u \cdot v \models r$, where r is a past augmented SERE and u, v are finite words means that v models tightly r given that the past is u. The semantics of past augmented SEREs are defined as follows, where b denotes a Boolean expression in $\mathsf{B}_{-\mathrm{LV}}$, and r, r_1, and r_2 denote past augmented SEREs:

1. $u \cdot v \models \{r\} \iff u \cdot v \models r$
2. $u \cdot v \models b \iff |v| = 1$ and $b(uv^0)$
3. $u \cdot v \models r_1 \ ; \ r_2 \iff \exists v_1, v_2$ s.t. $v = v_1v_2$, $u \cdot v_1 \models r_1$, and $uv_1 \cdot v_2 \models r_2$
4. $u \cdot v \models r_1 : r_2 \iff \exists v_1, v_2$, and σ s.t. $v = v_1\sigma v_2$, $u \cdot v_1\sigma \models r_1$, and $uv_1 \cdot \sigma v_2 \models r_2$
5. $u \cdot v \models r_1 \mid r_2 \iff u \cdot v \models r_1$ or $u \cdot v \models r_2$
6. $u \cdot v \models r_1 \ \&\& \ r_2 \iff u \cdot v \models r_1$ and $u \cdot v \models r_2$
7. $u \cdot v \models [*0] \iff v = \epsilon$
8. $u \cdot v \models r[*] \iff$ either $u \cdot v \models [*0]$ or $\exists v_1, v_2$ s.t. $v_1 \neq \epsilon$, $v = v_1v_2$, $u \cdot v_1 \models r$ and $uv_1 \cdot v_2 \models r[*]$

B.3.1.3.2 Past Augmented Semantics of FL Formulas

Let v be a possibly empty finite or infinite word over $\widehat{\Sigma}$, a be a Boolean expression in $\mathsf{B}_{-\mathrm{LV}}$, r a past augmented SERE, φ, ψ past augmented formulas, and n a non-negative integer. The notation $u \cdot v \models \varphi$ means that v models φ given that the past is u.

1. $u \cdot v \models (\varphi) \iff u \cdot v \models \varphi$
2. $u \cdot v \models \neg\varphi \iff \overline{u} \cdot \overline{v} \not\models \varphi$
3. $u \cdot v \models \varphi \wedge \psi \iff u \cdot v \models \varphi$ and $u \cdot v \models \psi$
4. $u \cdot v \models r! \iff \exists j < |v|$ s.t. $u \cdot v^{0..j} \models r$

5. $u \cdot v \models r \iff \forall j < |v|,\ u \cdot v^{0..j} \top^\omega \models r!$
6. $u \cdot v \models r \mapsto \varphi \iff \forall j < |v|$ s.t. $\overline{u} \cdot \overline{v}^{0..j} \not\models r$, $uv^{0..j-1} \cdot v^{j..} \models \varphi$
7. $u \cdot v \models \mathsf{X!}[n]\ \varphi \iff |v| > n$ and $uv^{0..n-1} \cdot v^{n..} \models \varphi$
8. $u \cdot v \models \varphi\ \mathsf{U}\ \psi \iff \exists k < |v|$ s.t. $uv^{0..k-1} \cdot v^{k..} \models \psi$, and $\forall j < k$, $uv^{0..j-1} \cdot v^{j..} \models \varphi$
9. $u \cdot v \models \varphi$ async_abort $a \iff$ either $u \cdot v \models \varphi$ or $\exists j < |v|$ s.t. $a(uv^{0..j})$=T and $u \cdot v^{0..j-1} \top^\omega \models \varphi$
10. $u \cdot v \models \varphi$ sync_abort $a \iff u \cdot v \models \varphi$ async_abort a

B.3.1.3.3 Past Augmented Semantics With Respect to a Model

Let φ be a past augmented formula and M a model. The notation $M \models \varphi$ means that for all π such that π is computation path of M, $\epsilon \cdot L(\pi) \models \varphi$.

B.3.1.4 Semantics with Local Variables

In this section we provide semantics of the base logic augmented with local variables and procedural blocks, which we refer to as the LVA *semantics*.

Let AP be a set of atomic propositions and LV a set of local variables with finite domain D. Let Σ be the set of all possible assignments of the atomic propositions AP (i.e. $\Sigma = 2^{AP}$) and Γ the set of all possible valuations of the local variables LV (i.e. $\Gamma = D^{LV}$). Let Λ denote the alphabet $\Sigma \times \Gamma \times \Gamma$. Let $\widehat{\Sigma}$, $\widehat{\Gamma}$ and $\widehat{\Lambda}$ denote the alphabets $\Sigma \cup \{\top, \bot\}$, $\Gamma \cup \{\top, \bot\}$ and $\Lambda \cup \{\mathbb{T}, \mathbb{\perp}\}$, respectively, where $\mathbb{T}, \mathbb{\perp}$ abbreviate $(\top, \top, \top)$ and $(\bot, \bot, \bot)$, respectively.

In the sequel we use u, v, w to denote words over $\widehat{\Sigma}$ and $\mathtt{u}, \mathtt{v}, \mathtt{w}$ to denote words over $\widehat{\Lambda}$. We use σ and $\mathtt{a}$ to denote a letter in $\widehat{\Sigma}$ and $\widehat{\Lambda}$, respectively.

Given a letter σ in $\widehat{\Sigma}$ (or γ in $\widehat{\Gamma}$) we use $\overline{\sigma}$ ($\overline{\gamma}$) to denote the *dual* of σ (γ) which is $\bot$ if σ (γ) is $\top$, is $\top$ if σ (γ) is $\bot$ and is σ (γ) otherwise. The definition of $\overline{\mathtt{a}}$ for $\mathtt{a} \in \widehat{\Lambda}$ is similar, replacing $\mathbb{T}$ with $\mathbb{\perp}$ and vice versa. We use $\overline{v}$ and $\overline{\mathtt{v}}$ to denote the obvious extensions of the definition of dual to words.

Let $\mathtt{v} = (\sigma_0, \gamma_0, \gamma_0')(\sigma_1, \gamma_1, \gamma_1') \cdots$ be a word over Λ. We use $\mathtt{v}|_\sigma$,$\mathtt{v}|_\gamma$,$\mathtt{v}|_{\gamma'}$ to denote the word obtained from $\mathtt{v}$ by leaving only the first, second or third component, respectively. That is $\mathtt{v}|_\sigma = \sigma_0\sigma_1 \cdots$, $\mathtt{v}|_\gamma = \gamma_0\gamma_1 \cdots$ and $\mathtt{v}|_{\gamma'} = \gamma_0'\gamma_1' \cdots$. The intuitive roles of the components of a letter are as follows. The first component of a letter $\mathtt{a}$ (i.e. $\mathtt{a}|_\sigma$) holds the valuation of the atomic propositions at the given cycle. The second component of a letter $\mathtt{a}$ (i.e. $\mathtt{a}|_\gamma$) holds the current values of the local variables at the given cycle (the *pre-value*). The third component of a letter $\mathtt{a}$ (i.e. $\mathtt{a}|_{\gamma'}$) holds the values of the local variables after the assignments have taken place (the *post-value*).

We say that $\mathtt{v}$ is *good* if for every $i \in \{0..|\mathtt{v}| - 2\}$ we have that either $\gamma_{i+1} \in \{\top, \bot\}$ or $\gamma_{i+1} = \gamma_i'$, that is, the pre-value of the local variables at letter $i + 1$ is the same as the post-value of letter i (i.e. is the result of the assignments that took place on letter i).

B.3.1.4.0 LVA Semantics of Expressions

The semantics of FL *formulas* is defined inductively, using as the base case the semantics of *expressions*. In the case of the LVA semantics, we do not have the built-in functions *ended*$(\cdot)$ and *prev*$(\cdot)$. We view an expression $e \in \mathsf{E}$ over $AP \cup LV$ as a mapping $e : \Sigma \times \Gamma \cup \{(\top, \top), (\bot, \bot)\} \mapsto D$ where D is the domain of variables in LV. Thus, $e(\sigma, \gamma)$ denotes the value of expression e under $\sigma \in \Sigma \cup \{\top, \bot\}$ and $\gamma \in \Gamma \cup \{\top, \bot\}$. For an extended letter $\mathtt{a}$ over $\widehat{\Lambda}$ we use $e(\mathtt{a})$ to abbreviate $e(\mathtt{a}|_\sigma, \mathtt{a}|_\gamma)$. We first discuss the case where $\mathtt{a} \in \Lambda$ (that is $\sigma \in \Sigma$ and $\gamma \in \Gamma$), then address the cases where $\mathtt{a} \in \{\mathbb{T}, \mathbb{\perp}\}$ (that is, $(\sigma, \gamma) \in \{(\top, \top), (\bot, \bot)\}$).

Let $\sigma \in \Sigma$ and $\gamma \in \Gamma$. When e is an atomic proposition, say e is p for some $p \in AP$, then $p(\sigma, \gamma)$ is equivalent to $p \in \sigma$. When e is a local variable, say e is z for some $z \in LV$, then $z(\sigma, \gamma)$ is the value given to z by γ. For an atomic proposition p, we abuse the notation by writing $p(\sigma)$ rather than $p(\sigma, \gamma)$. For a local variable z, we abuse the notation by writing $z(\gamma)$ rather than $z(\sigma, \gamma)$.

We assume T and F are in D. We define true and false to be the Boolean expressions that map every pair (σ, γ) where $\sigma \in \Sigma$ and $\gamma \in \Gamma$ to T and F, respectively.

We assume that operators are closed under D and that they behave in the usual manner, i.e. that for $\sigma \in \Sigma$, $\gamma \in \Gamma$, $e, e_1, e_2 \in \mathsf{E}$, a binary operator $\otimes$ and a unary operator $\circ$ we have $e_1(\sigma, \gamma) \otimes e_2(\sigma, \gamma) = (e_1 \otimes e_2)(\sigma, \gamma)$ and $\circ(e(\sigma, \gamma)) = (\circ e)(\sigma, \gamma)$. In particular we assume that Boolean disjunction, conjunction and negation behave in the usual manner.

Regarding the letters $\top$ and $\bot$, we assume any expression e is evaluated to T on $(\top, \top)$ and to F on $(\bot, \bot)$. That is $e(\top, \top) = \mathrm{T}$ and $e(\bot, \bot) = \mathrm{F}$ for any expression e. Note that in particular we have $\mathsf{true}(\top, \top) = \mathsf{false}(\top, \top) = \mathrm{T}$ and $\mathsf{true}(\bot, \bot) = \mathsf{false}(\bot, \bot) = \mathrm{F}$, and furthermore, even non-Boolean expressions are mapped to T and F from $(\top, \top)$ and $(\bot, \bot)$, respectively.

B.3.1.4.1 LVA Semantics of SEREs

The notation $\mathtt{v} \models_{\mathcal{Z}} r$, where r is an LVA SERE, $\mathtt{v}$ a finite word and $\mathcal{Z}$ a set of local variables means that $\mathtt{v}$ models tightly r under the set of controlled variables $\mathcal{Z}$. The semantics of LVA SEREs are defined as follows, where b denotes a Boolean expression in $\mathsf{B}_{\text{–PE}}$, and r, r_1, and r_2 denote LVA SEREs, z a local variable, $e, e_1, \ldots, e_n$ expressions over $LV \cup AP$, $\sigma \in \widehat{\Sigma}$ and $\gamma \in \widehat{\Gamma}$.

For a set of local variables $\mathcal{Y} = \{y_1, \ldots, y_n\}$, we denote by $\zeta(\mathcal{Y})$ a sequence of assignments $y_1 \leftarrow e_1, \ \ldots \ , y_n \leftarrow e_n$ to the variables in $\mathcal{Y}$, and define the valuation $[\zeta(\mathcal{Y})](\sigma, \gamma)$ recursively as follows. The base of the definition $[z \leftarrow e](\sigma, \gamma)$ is the valuation $\tilde{\gamma}$ such that $z(\tilde{\gamma}) = e(\sigma, \gamma)$ and for every local variable $y \in LV \setminus \{z\}$ we have $y(\tilde{\gamma}) = y(\gamma)$. In the step of the definition we have $[z \leftarrow e, \zeta(\mathcal{Y})](\sigma, \gamma) = [\zeta(\mathcal{Y})](\sigma, \tilde{\gamma})$ where $\tilde{\gamma} = [z \leftarrow e](\sigma, \gamma)$.

Let $\gamma_1, \gamma_2 \in \widehat{\Gamma}$ and $\mathcal{Z} \subseteq LV$. We say that γ_1 agrees with γ_2 relative to $\mathcal{Z}$, denoted $\gamma_1 \overset{\mathcal{Z}}{\sim} \gamma_2$, if for every $z \in \mathcal{Z}$ we have $z(\gamma_1) = z(\gamma_2)$.

1. $\mathtt{v} \models_{\mathcal{Z}} \{r\} \iff \mathtt{v} \models_{\mathcal{Z}} r$
2. $\mathtt{v} \models_{\mathcal{Z}} b \iff |\mathtt{v}| = 1$ and $b(\mathtt{v}^0)$=T and $\mathtt{v}^0|_{\gamma'} \overset{\mathcal{Z}}{\sim} \mathtt{v}^0|_{\gamma}$
3. $\mathtt{v} \models_{\mathcal{Z}} b, \zeta(\mathcal{Y}) \iff |\mathtt{v}| = 1$ and $b(\mathtt{v}^0)$=T and $\mathtt{v}^0|_{\gamma'} \overset{\mathcal{Z} \cup \mathcal{Y}}{\sim} [\zeta(\mathcal{Y})](\mathtt{v}^0)$
4. $\mathtt{v} \models_{\mathcal{Z}} r_1 \ ; \ r_2 \iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 \models_{\mathcal{Z}} r_1$, and $\mathtt{v}_2 \models_{\mathcal{Z}} r_2$
5. $\mathtt{v} \models_{\mathcal{Z}} r_1 : r_2 \iff \exists \mathtt{v}_1, \mathtt{v}_2, \mathtt{a}, \tilde{\gamma}$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{a}\mathtt{v}_2$, $\mathtt{v}_1\langle \mathtt{a}|_\sigma, \mathtt{a}|_\gamma, \tilde{\gamma}\rangle \models_{\mathcal{Z}} r_1$ and $\langle \mathtt{a}|_\sigma, \tilde{\gamma}, \mathtt{a}|_{\gamma'}\rangle \mathtt{v}_2 \models_{\mathcal{Z}} r_2$
6. $\mathtt{v} \models_{\mathcal{Z}} r_1 \mid r_2 \iff \mathtt{v} \models_{\mathcal{Z}} r_1$ or $\mathtt{v} \models_{\mathcal{Z}} r_2$
7. $\mathtt{v} \models_{\mathcal{Z}} r_1 \ \&\& \ r_2 \iff \mathtt{v} \models_{\mathcal{Z}} r_1$ and $\mathtt{v} \models_{\mathcal{Z}} r_2$
8. $\mathtt{v} \models_{\mathcal{Z}} [*0] \iff \mathtt{v} = \epsilon$
9. $\mathtt{v} \models_{\mathcal{Z}} r[*] \iff$ either $\mathtt{v} = \epsilon$ or $\exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v}_1 \neq \epsilon$, $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 \models_{\mathcal{Z}} r$ and $\mathtt{v}_2 \models_{\mathcal{Z}} r[*]$
10. $\mathtt{v} \models_{\mathcal{Z}} \{\mathsf{var}(z)\ r\} \iff \mathtt{v} \models_{\mathcal{Z} \cup \{z\}} r$
11. $\mathtt{v} \models_{\mathcal{Z}} \{\mathsf{free}(z)\ r\} \iff \mathtt{v} \models_{\mathcal{Z} \setminus \{z\}} r$

B.3.1.4.2 LVA Semantics of FL Formulas

Let w be a possibly empty finite or infinite word over $\widehat{\Sigma}$, a a basic Boolean expression in $\mathsf{B}_{-\mathrm{LV,PE}}$, r an LVA SERE, φ, ψ LVA formulas, and n a non-negative integer. Let $\gamma \in \widehat{\Gamma}$ be a valuation of the local variables and $\mathcal{Z} \subseteq LV$ a set of local variables. The notation $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi$ means that the word w satisfies φ under the set of local variables $\mathcal{Z}$ with the current valuation of variables γ.

Let w be a word over $\widehat{\Sigma}$, γ an element of $\widehat{\Gamma}$ and $\mathcal{Z}$ a set of local variables. We say that a word $\mathtt{w}$ over $\widehat{\Lambda}$ *enhances* $\langle w, \gamma\rangle$ iff $\mathtt{w}$ is good, $\mathtt{w}|_\sigma = w$, and if $\mathtt{w}^0_\sigma \notin \{\top, \bot\}$ then $\mathtt{w}^0|_\gamma = \gamma$.

1. $\langle w, \gamma\rangle \models_{\mathcal{Z}} (\varphi) \iff \langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi$
2. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \neg\varphi \iff \langle \overline{w}, \overline{\gamma}\rangle \not\models_{\mathcal{Z}} \varphi$
3. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi \wedge \psi \iff \langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi$ and $\langle w, \gamma\rangle \models_{\mathcal{Z}} \psi$
4. $\langle w, \gamma\rangle \models_{\mathcal{Z}} r! \iff \exists \mathtt{w}$ that enhances $\langle w, \gamma\rangle$ and $\exists j < |w|$ such that $\mathtt{w}^{0..j} \not\models_{\mathcal{Z}} r$
5. $\langle w, \gamma\rangle \models_{\mathcal{Z}} r \iff \forall j < |w|$, $\langle w^{0..j}\top^\omega, \gamma\rangle \models_{\mathcal{Z}} r!$
6. $\langle w, \gamma\rangle \models_{\mathcal{Z}} r \mapsto \varphi \iff \forall \mathtt{w}$ that enhances $\langle w, \gamma\rangle$ and $\forall j < |w|$: if $\overline{\mathtt{w}}^{0..j} \not\models_{\mathcal{Z}} r$ then $\langle w^{j..}, \mathtt{w}^j|_{\gamma'}\rangle \models_{\mathcal{Z}} \varphi$
7. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \mathsf{X!}[n]\ \varphi \iff |w| > n$ and $\langle w^{n..}, \gamma\rangle \models_{\mathcal{Z}} \varphi$
8. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi\ \mathsf{U}\ \psi \iff \exists k < |w|$ such that $\langle w^{k..}, \gamma\rangle \models_{\mathcal{Z}} \psi$ and $\forall j < k$, $\langle w^{j..}, \gamma\rangle \models_{\mathcal{Z}} \varphi$
9. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi\ \mathsf{async\_abort}\ a \iff$ either $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi$ or $\exists j < |w|$ s.t. $a(w^j)$=T and $\langle w^{0..j-1}\top^\omega, \gamma\rangle \models_{\mathcal{Z}} \varphi$
10. $\langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi\ \mathsf{sync\_abort}\ a \iff \langle w, \gamma\rangle \models_{\mathcal{Z}} \varphi\ \mathsf{async\_abort}\ a$
11. $\langle w, \gamma\rangle \models_{\mathcal{Z}} (\mathsf{var}(z)\ \varphi) \iff \langle w, \gamma\rangle \models_{\mathcal{Z} \cup \{z\}} \varphi$

B.3.1.4.3 LVA Semantics With Respect to a Model

Let φ be an LVA formula and M a model. The notation $M \models \varphi$ means that for all π such that π is computation path of M, $L(\pi) \models_\emptyset \varphi$.

B.3.1.5 Semantics with Everything

In this section we provide semantics of FL formulas. We use the same alphabet and notation as in the LVA semantics, and in addition we define a *clock tick* as follows.

Let c be a Boolean expression in $\mathsf{B}_{-\mathrm{LV}}$. We say that finite word $\mathtt{v}$ with past $\mathtt{u}$ *is a clock tick of* c, denoted $tick_c(\mathtt{u} \cdot \mathtt{v})$, if $|\mathtt{v}| > 0$ and $c(\mathtt{u}\mathtt{v}^{0..|\mathtt{v}|-1})$=T and for every non-negative integer $i < |\mathtt{v}| - 1$, $c(\mathtt{u}\mathtt{v}^{0..i})$=F. For an integer $m > 0$ we say that finite word $\mathtt{v}$ with past $\mathtt{u}$ *is* m *clock ticks of* c, denoted $ticks_c(\mathtt{u} \cdot \mathtt{v}, m)$, if there exist m words $\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_m$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2 \cdots \mathtt{v}_m$ and for every $1 \le i \le m$ we have $tick_c(\mathtt{u}\mathtt{v}_1 \cdots \mathtt{v}_{i-1} \cdot \mathtt{v}_i)$.

We say that finite word $\mathtt{v}$ with past $\mathtt{u}$ *is a past clock tick of* c, denoted $past\_tick_c(\mathtt{u} \cdot \mathtt{v})$, if $|\mathtt{v}| > 0$ and $c(\mathtt{u}\mathtt{v}^0)$=T and for every non-negative integer $0 < i < |\mathtt{v}|$, $c(\mathtt{u}\mathtt{v}^{0..i})$=F (where $c(\cdot)$ refers to the base semantics of expressions). For an integer $m > 0$ we say that finite word $\mathtt{v}$ with past $\mathtt{u}$ *is* m *past clock ticks of* c, denoted $past\_ticks_c(\mathtt{u} \cdot \mathtt{v}, m)$, if there exist m words $\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_m$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2 \cdots \mathtt{v}_m$ and for every $1 \le i \le m$ we have $past\_tick_c(\mathtt{u}\mathtt{v}_1 \cdots \mathtt{v}_{i-1} \cdot \mathtt{v}_i)$.

B.3.1.5.0 Semantics of Expressions

The semantics of expressions (that may refer to the past) over $AP \cup LV$ maps non-empty words over $\widehat{\Lambda}$ and a clock context to the domain D of local variables. Let $\mathtt{u}$ and $\mathtt{a}$ be a finite word and a letter over $\widehat{\Lambda}$, respectively. Let c be a boolean expression in $\mathsf{B}_{-\mathrm{LV}}$. The semantics of expression e under clock context c, denoted $e_c(\cdot)$, is defined as follows (where $e(\mathtt{a})$ below refers to the LVA semantics of expressions).

- If $e \in \mathsf{E}$ is an expression without $prev(\cdot)$ or $ended(\cdot)$ then $e_c(\mathtt{ua}) = e(\mathtt{a})$
- If r is a SERE and c, c_1 are Boolean expressions then
  - $ended_c(r, c_1)(\mathtt{ua})$=T iff there exist $\mathtt{u}_1, \mathtt{u}_2$ such that $\mathtt{u} = \mathtt{u}_1\mathtt{u}_2$ and $\mathtt{u}_1 \cdot \mathtt{u}_2\mathtt{a} \models^{c_1}_{\emptyset} r$
  - $ended_c(r)(\mathtt{ua}) = ended_c(r, c)(\mathtt{ua})$
- If e is an expression and c, c_1 Boolean expressions then
  - $prev_c(e, n, c_1)(\mathtt{u}) = \begin{cases} e_{c_1}(\mathtt{u}^{0..i}) & \text{if } \exists i < |\mathtt{u}| \text{ such that } past\_ticks_{c_1}(\mathtt{u}^{0..i-1} \cdot \mathtt{u}^{i..|u|-1}, n+1) \\ \text{undefined} & \text{otherwise.} \end{cases}$
  - $prev_c(e, n)(\mathtt{u}) = prev_c(e, n, c)(\mathtt{u})$
  - $prev_c(e)(\mathtt{u}) = prev_c(e, 1, c)(\mathtt{u})$

Note that $prev(e, n, c)(\mathtt{u})$ (as well as its derivatives as given in Section B.4.0) is undefined when $\mathtt{u}$ does not contain $n+1$ letters on which c holds.

B.3.1.5.1 Semantics of SEREs

Let r be a SERE, $\mathtt{u}, \mathtt{v}$ finite words over $\widehat{\Lambda}$, $\mathcal{Z} \subseteq LV$ a set of local variables and c a Boolean expression in $\mathsf{B}_{(LV)}$. The notation $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r$ means that v models tightly r under the set of controlled variables $\mathcal{Z}$ and clock context c assuming the past is $\mathtt{u}$. The semantics of SEREs are defined as follows, where b denotes a Boolean expression in B, c, c_1 denote Boolean expressions in $\mathsf{B}_{(LV)}$, and r, r_1, and r_2 denote SEREs.

For a set of local variables $\mathcal{Y} = \{y_1, \ldots, y_n\}$, we denote by $\zeta(\mathcal{Y})$ a sequence of assignments $y_1 \leftarrow e_1, \ \ldots \ , y_n \leftarrow e_n$ to the variables in $\mathcal{Y}$, and define the valuation $[\zeta(\mathcal{Y})]_c(\mathtt{ua})$ recursively as follows. The base of the definition $[z \leftarrow e]_c(\mathtt{ua})$ is the valuation $\tilde{\gamma}$ such that $z(\tilde{\gamma}) = e_c(\mathtt{ua})$ and for every local variable $y \in LV \setminus \{z\}$ we have $y(\tilde{\gamma}) = y(\mathtt{a}|_\gamma)$. In the step of the definition we have $[z \leftarrow e, \zeta(\mathcal{Y})]_c(\mathtt{ua}) = [\zeta(\mathcal{Y})]_c(\mathtt{u}\langle \mathtt{a}|_\sigma, \tilde{\gamma}, \mathtt{a}|_{\gamma'} \rangle)$ where $\tilde{\gamma} = [z \leftarrow e]_c(\mathtt{ua})$.

Let $\gamma_1, \gamma_2 \in \widehat{\Gamma}$ and $\mathcal{Z} \subseteq LV$. We say that γ_1 agrees with γ_2 relative to $\mathcal{Z}$, denoted $\gamma_1 \overset{\mathcal{Z}}{\sim} \gamma_2$, if for every $z \in \mathcal{Z}$ we have $z(\gamma_1) = z(\gamma_2)$.

1. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} \{r\} \iff \mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r$
2. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} b \iff$
   $tick_c(\mathtt{u} \cdot \mathtt{v})$ and $b_c(\mathtt{uv})$=T and $\forall i < |\mathtt{v}|$ we have $\mathtt{v}^i|_{\gamma'} \overset{\mathcal{Z}}{\sim} \mathtt{v}^i|_\gamma$
3. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} b, \zeta(\mathcal{Y}) \iff$
   $tick_c(\mathtt{u} \cdot \mathtt{v})$ and $b_c(\mathtt{uv})$=T and $\forall i < |\mathtt{v} - 1|$ we have $\mathtt{v}^i|_{\gamma'} \overset{\mathcal{Z}}{\sim} \mathtt{v}^i|_\gamma$ and $\mathtt{v}^{|\mathtt{v}|-1}|_{\gamma'} \overset{\mathcal{Z} \cup \mathcal{Y}}{\sim} [\zeta(\mathcal{Y})]_c(\mathtt{uv})$
4. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1 \ ; \ r_2 \iff$
   $\exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{u} \cdot \mathtt{v}_1 \models^{c}_{\mathcal{Z}} r_1$, and $\mathtt{u}\mathtt{v}_1 \cdot \mathtt{v}_2 \models^{c}_{\mathcal{Z}} r_2$
5. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1 : r_2 \iff$
   $\exists \mathtt{v}_1, \mathtt{v}_2, \mathtt{a}, \tilde{\gamma}$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{a}\mathtt{v}_2$, $\mathtt{u} \cdot \mathtt{v}_1\langle \mathtt{a}|_\sigma, \mathtt{a}|_\gamma, \tilde{\gamma} \rangle \models^{c}_{\mathcal{Z}} r_1$ and $\mathtt{u}\mathtt{v}_1 \cdot \langle \mathtt{a}|_\sigma, \tilde{\gamma}, \mathtt{a}|_{\gamma'} \rangle \mathtt{v}_2 \models^{c}_{\mathcal{Z}} r_2$
6. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1 \mid r_2 \iff$
   $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1$ or $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_2$
7. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1 \ \&\& \ r_2 \iff$
   $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_1$ and $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r_2$
8. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} [*0] \iff$
   $\mathtt{v} = \epsilon$

9. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r[*] \iff$
   either $\mathtt{v} = \epsilon$ or $\exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v}_1 \neq \epsilon$, $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{u} \cdot \mathtt{v}_1 \models^{c}_{\mathcal{Z}} r$ and $\mathtt{u}\mathtt{v}_1 \cdot \mathtt{v}_2 \models^{c}_{\mathcal{Z}} r[*]$
10. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} r@c_1 \iff \mathtt{u} \cdot \mathtt{v} \models^{c_1}_{\mathcal{Z}} r$
11. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} \{\mathsf{var}(z)\ r\} \iff$
    $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z} \cup \{z\}} r$
12. $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z}} \{\mathsf{free}(z)\ r\} \iff$
    $\mathtt{u} \cdot \mathtt{v} \models^{c}_{\mathcal{Z} \setminus \{z\}} r$

B.3.1.5.2 Semantics of FL Formulas

Let $\mathtt{u}$ be a possibly empty finite word over $\widehat{\Lambda}$, w a possibly empty finite or infinite word over $\widehat{\Sigma}$, b a Boolean expression in B, a, c, c_1 Boolean expressions in $\mathsf{B}_{-\mathrm{LV}}$, r a SERE, φ, ψ FL formulas, and n a non-negative integer. Let $\gamma \in \widehat{\Gamma}$ and $\mathcal{Z} \subseteq LV$ a set of local variables. The notation $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$ means that the word w satisfies φ under the set of local variables $\mathcal{Z}$ with the current valuation of variables γ given the past is $\mathtt{u}$ and the clock context is c.

Let w be a word over $\widehat{\Sigma}$, γ an element of $\widehat{\Gamma}$ and $\mathcal{Z}$ a set of local variables. We say that a word $\mathtt{w}$ over $\widehat{\Lambda}$ *enhances* $\langle w, \gamma \rangle$ iff $\mathtt{w}$ is good, $\mathtt{w}|_\sigma = w$, and if $\mathtt{w}^0_\sigma \notin \{\top, \bot\}$ then $\mathtt{w}^0|_\gamma = \gamma$. We use $\langle\!\langle w, \gamma \rangle\!\rangle$ to denote the word enhancing $\langle w, \gamma \rangle$ such that for every $j < |w|$ we have $\langle\!\langle w, \gamma \rangle\!\rangle^j|_\gamma = \langle\!\langle w, \gamma \rangle\!\rangle^j|_{\gamma'} \in \{\gamma, \top, \bot\}$.

1. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} (\varphi) \iff \mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$
2. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \neg\varphi \iff$
   $\overline{\mathtt{u}} \cdot \langle \overline{w}, \overline{\gamma} \rangle \not\models^{\mathcal{G}}_{\mathcal{Z}} \varphi$
3. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi \wedge \psi \iff$
   $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$ and $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \psi$
4. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} r! \iff$
   $\exists \mathtt{w}$ that enhances $\langle w, \gamma \rangle$ and $\exists j < |w|$ such that $\mathtt{u} \cdot \mathtt{w}^{0..j} \models^{c}_{\mathcal{Z}} r$
5. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} r \iff$
   $\forall j < |w|$, $\mathtt{u} \cdot \langle w^{0..j}\top^\omega, \gamma \rangle \models^{c}_{\mathcal{Z}} r!$
6. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} r \mapsto \varphi \iff$
   $\forall \mathtt{w}$ that enhances $\langle w, \gamma \rangle$ and $\forall j < |w|$: if $\overline{\mathtt{u}} \cdot \overline{\mathtt{w}}^{0..j} \models^{c}_{\mathcal{Z}} r$ then $\mathtt{u}\mathtt{w}^{0..j-1} \cdot \langle w^{j..}, \mathtt{w}^j|_{\gamma'} \rangle \models^{c}_{\mathcal{Z}} \varphi$
7. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \mathsf{X}![n]\ \varphi \iff$
   $\exists j < |w|$ s.t. $ticks_c(\mathtt{u} \cdot \langle\!\langle w, \gamma \rangle\!\rangle^{0..j}, n+1)$ and $\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^{0..j-1} \cdot \langle w^{j..}, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$
8. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi\ \mathsf{U}\ \psi \iff$
   $\exists k < |w|$ such that $c(\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^{0..k})$=T and $\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^{0..k-1} \cdot \langle w^{k..}, \gamma \rangle \models^{c}_{\mathcal{Z}} \psi$ and $\forall j < k$ such that $c(\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^{0..j})$=T we have $\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^{0..j-1} \cdot \langle w^{j..}, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$
9. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi@c_1 \iff$
   $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c_1}_{\mathcal{Z}} \varphi$
10. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi\ \mathsf{async\_abort}\ a \iff$
    either $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$ or $\exists j < |w|$ s.t. $a_c(\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^j)$=T and $\mathtt{u} \cdot \langle w^{0..j-1}\top^\omega, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$
11. $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi\ \mathsf{sync\_abort}\ a \iff$
    either $\mathtt{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$ or $\exists j < |w|$ s.t. $(a \wedge c)_c(\mathtt{u}\langle\!\langle w, \gamma \rangle\!\rangle^j)$=T and $\mathtt{u} \cdot \langle w^{0..j-1}\top^\omega, \gamma \rangle \models^{c}_{\mathcal{Z}} \varphi$

12. $\mathfrak{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z}} (\mathsf{var}(z)\ \varphi) \iff$
$\mathfrak{u} \cdot \langle w, \gamma \rangle \models^{c}_{\mathcal{Z} \cup \{z\}} \varphi$

B.3.1.5.3 Semantics With Respect to a Model

Let φ be an FL formula and M a model. The notation $M \models \varphi$ means that for all π such that π is computation path of M, and for all $\gamma \in \Gamma$ we have $\epsilon \cdot \langle L(\pi), \gamma \rangle \models^{\mathsf{true}}_{\emptyset} \varphi$.

B.3.2 Semantics of OBE formulas

The semantics of OBE formulas are defined over states in the *model*, rather than finite or infinite words. Let f be an OBE formula, $M = (S, S_0, R, P, L)$ a model and $s \in S$ a state of the model. The notation $M, s \models f$ means that 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 OBE formulas are defined inductively, using as the base case the semantics of *Boolean expressions* over *letters* in 2^P. The semantics of Boolean expression is assumed to be given as a relation $\Vdash\ \subseteq 2^P \times \mathsf{B}_{-\mathrm{LV,PE}}$ relating letters in 2^P with Boolean expressions in $\mathsf{B}_{-\mathrm{LV,PE}}$. If $(\ell, b) \in \Vdash$ we say that the letter ℓ *satisfies* the Boolean expression b and denote it $\ell \Vdash b$. We assume that the Boolean relation $\Vdash$ behaves in the usual manner. In particular, that for every letter $\ell \in 2^P$, atomic proposition $p \in P$ and Boolean expressions $b, b_1, b_2 \in \mathsf{B}_{-\mathrm{LV,PE}}$ (i) $\ell \Vdash p$ iff $p \in \ell$, (ii) $\ell \Vdash \neg b$ iff $\ell \nVdash b$, (iii) $\ell \Vdash b_1 \wedge b_2$ iff $\ell \Vdash b_1$ and $\ell \Vdash b_2$, and (iv) $\ell \Vdash \mathsf{true}$ and $\ell \nVdash \mathsf{false}$.

The semantics of an OBE formula are those of standard CTL. The semantics are defined as follows, where b denotes a Boolean expression and f, f_1, and f_2 denote OBE formulas.

1. $M, s \models b \iff L(s) \Vdash b$
2. $M, s \models (f) \iff M, s \models f$
3. $M, s \models \neg f \iff M, s \not\models f$
4. $M, s \models f_1 \wedge f_2 \iff M, s \models f_1$ and $M, s \models f_2$
5. $M, s \models \mathsf{EX}\ f \iff$ there exists a computation path π of M such that $|\pi| > 1$, $\pi_0 = s$, and $M, \pi_1 \models f$
6. $M, s \models \mathsf{E}[f_1\ \mathsf{U}\ f_2] \iff$ 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 $j < k$: $M, \pi_j \models f_1$
7. $M, s \models \mathsf{EG}\ f \iff$ there exists a computation path π of M such that $\pi_0 = s$ and for every j such that $0 \leq j < |\pi|$: $M, \pi_j \models f$

B.4 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 operators presented above. The syntactic sugar is defined below.

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.

B.4.1 Additional Boolean expressions

Let $b \in \mathsf{B}$ and $e \in \mathsf{E}$. Then additional Boolean operators can be viewed as abbreviations of the basic Boolean operators given in Definition 1, as follows:

- $rose(b) \stackrel{\text{def}}{=} \neg prev(b) \wedge b$
- $rose(b, c) \stackrel{\text{def}}{=} \neg prev(b, c) \wedge b \wedge c$
- $fell(b) \stackrel{\text{def}}{=} prev(b) \wedge \neg b$
- $fell(b, c) \stackrel{\text{def}}{=} prev(b, c) \wedge \neg b \wedge c$
- $stable(e) \stackrel{\text{def}}{=} prev(e) = e$
- $stable(e, c) \stackrel{\text{def}}{=} prev(e, c) = e \wedge c$

B.4.2 Additional SEREs

We regard a *procedural block* as mechanism for both calculating a set of expressions and assigning them to a set of local variables which are given to the procedural block as inputs. Formally,

- $f(z_1, \ldots, z_n)$ where f is a procedural block and $z_1, \ldots, z_n$ are a local variables is interpreted as an abbreviation to the sequence of assignments $z_1 \leftarrow e_{f_1}, \ldots, z_n \leftarrow e_{f_n}$ where $e_{f_1}, \ldots, e_{f_n}$ are expressions corresponding to intermediate calculations of f.

B.4.3 Additional SERE operators

Let i, j, k, and l be integer constants such that $i \geq 0$, $j \geq i$, $k \geq 1$, $l \geq k$. Then additional SERE operators can be viewed as abbreviations of the basic SERE operators given in Definition 2, as follows, where b denotes a boolean expression, and r denotes a SERE.

- $r[+] \stackrel{\text{def}}{=} r; r[*]$
- $r[*0] \stackrel{\text{def}}{=} [*0]$
- $r[*k] \stackrel{\text{def}}{=} \overbrace{r; r; \ldots; r}^{k \ times}$
- $r[*i..j] \stackrel{\text{def}}{=} r[*i] \mid \ldots \mid r[*j]$
- $r[*i..] \stackrel{\text{def}}{=} r[*i]; r[*]$
- $r[*..i] \stackrel{\text{def}}{=} r[*0] \mid \ldots \mid r[*i]$
- $r[*..] \stackrel{\text{def}}{=} r[*0..]$
- $[+] \stackrel{\text{def}}{=} \mathsf{true}[+]$
- $[*] \stackrel{\text{def}}{=} \mathsf{true}[*]$
- $[*i] \stackrel{\text{def}}{=} \mathsf{true}[*i]$
- $[*i..j] \stackrel{\text{def}}{=} \mathsf{true}[*i..j]$
- $[*i..] \stackrel{\text{def}}{=} \mathsf{true}[*i..]$
- $[*..i] \stackrel{\text{def}}{=} \mathsf{true}[*..i]$

- $[*..] \stackrel{\text{def}}{=} \mathsf{true}[*..]$
- $b[=i] \stackrel{\text{def}}{=} \{\neg b[*]; b\}[*i]; \neg b[*]$
- $b[=i..j] \stackrel{\text{def}}{=} b[=i] \mid ... \mid b[=j]$
- $b[=i..] \stackrel{\text{def}}{=} b[=i]; [*]$
- $b[=..i] \stackrel{\text{def}}{=} b[=0] \mid ... \mid b[=i]$
- $b[=..] \stackrel{\text{def}}{=} b[=0..]$
- $b[\rightarrow] \stackrel{\text{def}}{=} \neg b[*]; b$
- $b[\rightarrow k] \stackrel{\text{def}}{=} \{\neg b[*]; b\}[*k]$
- $b[\rightarrow k..l] \stackrel{\text{def}}{=} b[\rightarrow k] \mid ... \mid b[\rightarrow l]$
- $b[\rightarrow k..] \stackrel{\text{def}}{=} b[\rightarrow k] \mid \{b[\rightarrow k]; [*]; b\}$
- $b[\rightarrow ..k] \stackrel{\text{def}}{=} b[\rightarrow 1] \mid ... \mid b[\rightarrow k]$
- $b[\rightarrow ..] \stackrel{\text{def}}{=} b[\rightarrow 1..]$
- $\mathsf{skip} \stackrel{\text{def}}{=} \{\mathsf{free}(LV)\ \mathsf{true}\}$
- $r_1\ \&\ r_2 \stackrel{\text{def}}{=} \{\ \{r_1; \mathsf{skip}[*]\}\ \&\&\ r_2\ \} \mid \{\ r_1\ \&\&\ \{r_2; \mathsf{skip}[*]\}\ \}$
- $r_1\ within\ r_2 \stackrel{\text{def}}{=} \{[*];\ r_1;\ [*]\}\ \&\&\ \{r_2\}$ Check it now that we have local variables free etc.
- $\{\mathsf{var}(z \leftarrow e)\ r\} \stackrel{\text{def}}{=} \{\mathsf{var}(z)\ \{\{\mathsf{true}, z \leftarrow e\} : r\} \mid \{[*0]\ \&\&\ r\}\}$

B.4.4 Additional FL operators

Let i, j, k and l are integers such that $i \geq 0$, $j \geq i$, $k > 0$ and $l \geq k$. Then additional FL operators can be viewed as abbreviations of the basic operators given in Definition 3, as follows, where b denotes a boolean expression, r, r_1, and r_2 denote SEREs, and φ, φ_1, and φ_2 denote FL formulas.

- $\varphi_1 \vee \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1 \rightarrow \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2$
- $\varphi_1 \leftrightarrow \varphi_2 \stackrel{\text{def}}{=} (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$

- $\mathsf{X!}\varphi \stackrel{\text{def}}{=} \mathsf{X!}[1]\ \varphi$
- $\mathsf{X}\varphi \stackrel{\text{def}}{=} \mathsf{X}[1]\ \varphi$

- $\mathsf{X}[i]\varphi \stackrel{\text{def}}{=} \neg\mathsf{X!}[i]\ \neg\varphi$
- $\mathsf{F}\varphi \stackrel{\text{def}}{=} \mathsf{true}\ \mathsf{U}\ \varphi$
- $\mathsf{G}\varphi \stackrel{\text{def}}{=} \neg\mathsf{F}\neg\varphi$
- $\varphi_1\ \mathsf{W}\ \varphi_2 \stackrel{\text{def}}{=} (\varphi_1\ \mathsf{U}\ \varphi_2) \vee \mathsf{G}\varphi_1$

- $always\ \varphi \stackrel{\text{def}}{=} \mathsf{G}\ \varphi$
- $never\ \varphi \stackrel{\text{def}}{=} \mathsf{G}\ \neg\varphi$
- $next!\ \varphi \stackrel{\text{def}}{=} \mathsf{X!}\ \varphi$
- $next\ \varphi \stackrel{\text{def}}{=} \mathsf{X}\ \varphi$
- $eventually!\ \varphi \stackrel{\text{def}}{=} \mathsf{F}\varphi$

- $\varphi_1\ until!\ \varphi_2 \stackrel{\text{def}}{=} \varphi_1\ \mathsf{U}\ \varphi_2$
- $\varphi_1\ until\ \varphi_2 \stackrel{\text{def}}{=} \varphi_1\ \mathsf{W}\ \varphi_2$
- $\varphi_1\ until!\_\ \varphi_2 \stackrel{\text{def}}{=} \varphi_1\ \mathsf{U}\ (\varphi_1 \wedge \varphi_2)$
- $\varphi_1\ until\_\ \varphi_2 \stackrel{\text{def}}{=} \varphi_1\ \mathsf{W}\ (\varphi_1 \wedge \varphi_2)$

- $\varphi_1\ before!\ \varphi_2 \stackrel{\text{def}}{=} (\neg\varphi_2)\ \mathsf{U}\ (\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1\ before\ \varphi_2 \stackrel{\text{def}}{=} (\neg\varphi_2)\ \mathsf{W}\ (\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1\ before!\_\ \varphi_2 \stackrel{\text{def}}{=} (\neg\varphi_2)\ \mathsf{U}\ \varphi_1$
- $\varphi_1\ before\_\ \varphi_2 \stackrel{\text{def}}{=} (\neg\varphi_2)\ \mathsf{W}\ \varphi_1$

- $next![i]\ \varphi \stackrel{\text{def}}{=} \mathsf{X!}[i]\ \varphi$
- $next[i]\ \varphi \stackrel{\text{def}}{=} \mathsf{X}[i]\ \varphi$
- $next\_a![i..j]\varphi \stackrel{\text{def}}{=} (\mathsf{X!}[i]\varphi) \wedge \ldots \wedge (\mathsf{X!}[j]\varphi)$
- $next\_a[i..j]\varphi \stackrel{\text{def}}{=} (\mathsf{X}[i]\varphi) \wedge \ldots \wedge (\mathsf{X}[j]\varphi)$
- $next\_e![i..j]\varphi \stackrel{\text{def}}{=} (\mathsf{X!}[i]\varphi) \vee \ldots \vee (\mathsf{X!}[j]\varphi)$
- $next\_e[i..j]\varphi \stackrel{\text{def}}{=} (\mathsf{X}[i]\varphi) \vee \ldots \vee (\mathsf{X}[j]\varphi)$

- $next\_event!(b)(\varphi) \stackrel{\text{def}}{=} (\neg b)\ \mathsf{U}\ b \wedge \varphi$
- $next\_event(b)(\varphi) \stackrel{\text{def}}{=} (\neg b)\ \mathsf{W}\ (b \wedge \varphi)$
- $next\_event!(b)[k](\varphi) \stackrel{\text{def}}{=} next\_event!(b)\ \overbrace{(\mathsf{X!}\ next\_event!(b)\ldots(\mathsf{X!}\ next\_event!(b)}^{k-1\ times}(\varphi))\ldots)$
- $next\_event(b)[k](\varphi) \stackrel{\text{def}}{=} next\_event(b)\ \overbrace{(\mathsf{X}next\_event(b)\ldots(\mathsf{X}next\_event(b)}^{k-1\ times}(\varphi))\ldots)$
- $next\_event\_a!(b)[k..l](\varphi) \stackrel{\text{def}}{=} next\_event!(b)[k](\varphi) \wedge \ldots \wedge next\_event!(b)[l](\varphi)$
- $next\_event\_a(b)[k..l](\varphi) \stackrel{\text{def}}{=} next\_event(b)[k](\varphi) \wedge \ldots \wedge next\_event(b)[l](\varphi)$
- $next\_event\_e!(b)[k..l](\varphi) \stackrel{\text{def}}{=} next\_event!(b)[k](\varphi) \vee \ldots \vee next\_event!(b)[l](\varphi)$

- $next\_event\_e(b)[k..l](\varphi) \stackrel{\text{def}}{=} next\_event(b)[k](\varphi) \vee ... \vee next\_event(b)[l](\varphi)$

- $r(\varphi) \stackrel{\text{def}}{=} r \mapsto \varphi$
- $r \Mapsto \varphi \stackrel{\text{def}}{=} \{r; \mathsf{true}\} \mapsto \varphi$

- φ abort $b \stackrel{\text{def}}{=} \varphi$ async_abort b

B.4.5 Parameterized SEREs and Formulas

Let r be a SERE, and l, m integers. Let S be a set of constants, integers or boolean values and p an identifier. The left hand side of the following are SEREs, equivalent to the SEREs on the right hand side.

- for p *in* S : | $r \stackrel{\text{def}}{=} \underset{s\in S}{|} \{r[p \leftarrow s]\}$.
- for $p\langle l..m\rangle$ *in* S : | $r \stackrel{\text{def}}{=} \underset{s_l\in S}{|} \ldots \underset{s_m\in S}{|} \{r[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]\}$
- for p *in* S : && $r \stackrel{\text{def}}{=} \underset{s\in S}{\&\&} \{r[p \leftarrow s]\}$.
- for $p\langle l..m\rangle$ *in* S : && $r \stackrel{\text{def}}{=} \underset{s_l\in S}{\&\&} \ldots \underset{s_m\in S}{\&\&} \{r[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]\}$
- for p *in* S : & $r \stackrel{\text{def}}{=} \underset{s\in S}{\&} \{r[p \leftarrow s]\}$.
- for & $p\langle l..m\rangle$ *in* S : & $r \stackrel{\text{def}}{=} \underset{s_l\in S}{\&} \ldots \underset{s_m\in S}{\&} \{r[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]\}$

Where $r[p \leftarrow s]$ is the SERE obtained from r by replacing every occurrence of p by s and $r[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]$ is the SERE obtained from r by replacing every occurrence of p_j with s_j for all j such that $l \leq j \leq m$.

Let φ be an FL formula, and l, m integers. Let S be a set of constants, integers or boolean values and p an identifier. The left hand side of the following are FL formulas equivalent to the FL formulas on the right hand side.

- for p *in* $S : \vee\ \varphi \stackrel{\text{def}}{=} \bigvee_{s\in S} \varphi[p \leftarrow s]$
- for $p\langle l..m\rangle$ *in* $S : \vee\ \varphi \stackrel{\text{def}}{=} \bigvee_{s_l\in S} \ldots \bigvee_{s_m\in S} \varphi[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]$
- for p *in* $S : \wedge\ \varphi \stackrel{\text{def}}{=} \bigwedge_{s\in S} \varphi[p \leftarrow s]$
- for $p\langle l..m\rangle$ *in* $S : \wedge\ \varphi \stackrel{\text{def}}{=} \bigwedge_{s_l\in S} \ldots \bigwedge_{s_m\in S} \varphi[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]$
- forall p *in* S : $\varphi \stackrel{\text{def}}{=}$ for p *in* $S : \wedge\ \varphi$
- forall $p\langle l..m\rangle$ *in* S : $\varphi \stackrel{\text{def}}{=}$ for $p\langle l..m\rangle$ *in* $S : \wedge\ \varphi$

where $\varphi[p \leftarrow s]$ is the formula obtained from φ by replacing every occurrence of p by s and $\varphi[p\langle l..m\rangle \leftarrow \langle s_l..s_m\rangle]$ is the formula obtained from φ by replacing every occurrence of p_j with s_j for all j such that $l \leq j \leq m$.

B.4.6 Additional OBE operators

Let f, f_1, f_2 denote OBE formulas. Then additional OBE operators can be derived from the basic OBE operators given in Definition 4 as follows:

- $f_1 \vee f_2 = \neg(\neg f_1 \wedge \neg f_2)$
- $f_1 \rightarrow f_2 = \neg f_1 \vee f_2$
- $f_1 \leftrightarrow f_2 = (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$
- $\mathsf{EF} f = \mathsf{E}[\mathsf{true}\ \mathsf{U}\ f]$
- $\mathsf{AX} f = \neg\mathsf{EX}\neg f$
- $\mathsf{A}[f_1\ \mathsf{U}\ f_2] = \neg(\mathsf{E}[\neg f_2\ \mathsf{U}\ (\neg f_1 \wedge \neg f_2)] \vee \mathsf{EG}\neg f_2)$
- $\mathsf{AG} f = \neg\mathsf{E}[\mathsf{true}\ \mathsf{U}\ \neg f]$
- $\mathsf{AF} f = \mathsf{A}[\mathsf{true}\ \mathsf{U}\ f]$

B.5 Rewriting rules for clocks

In Section B.3.1.2 we gave the semantics of clocked FL formulas directly. There is an equivalent definition in terms of FL formulas without clocks, as follows: Starting from the outermost clock, use the following rules to translate clocked SEREs into unclocked SEREs, and clocked FL formulas into unclocked FL formulas.

The rewrite rules for SEREs are:

1. $\mathcal{R}^c(\{r\}) = \mathcal{R}^c(r)$
2. $\mathcal{R}^c(b) = \neg c[*]; c \wedge b$
3. $\mathcal{R}^c(b, \zeta(\mathcal{Y})) = \neg c[*]; c \wedge b, \zeta(\mathcal{Y})$
4. $\mathcal{R}^c(r_1\ ;\ r_2) = \mathcal{R}^c(r_1)\ ; \mathcal{R}^c(r_2)$
5. $\mathcal{R}^c(r_1 : r_2) = \{\mathcal{R}^c(r_1)\} : \{\mathcal{R}^c(r_2)\}$
6. $\mathcal{R}^c(r_1\ \ |\ \ r_2) = \{\mathcal{R}^c(r_1)\}\ \ |\ \ \{\mathcal{R}^c(r_2)\}$
7. $\mathcal{R}^c(r_1\ \ \&\&\ \ r_2) = \{\mathcal{R}^c(r_1)\}\ \ \&\&\ \ \{\mathcal{R}^c(r_2)\}$
8. $\mathcal{R}^c([*0]) = [*0]$
9. $\mathcal{R}^c(r[*]) = \{\mathcal{R}^c(r)\}[*]$
10. $\mathcal{R}^c(\mathsf{var}(z)\ r) = \{\mathsf{var}(z)\ \mathcal{R}^c(r)\}$
11. $\mathcal{R}^c(\mathsf{free}(z)\ r) = \{\mathsf{free}(z)\ \mathcal{R}^c(r)\}$
12. $\mathcal{R}^c(r@c_1) = \mathcal{R}^{c_1}(r)$

The rewrite rules for FL formulas are:

1. $\mathcal{F}^c((\varphi)) = (\mathcal{F}^c(\varphi))$

2. $\mathcal{F}^c(\neg\varphi) = \neg\mathcal{F}^c(\varphi)$
3. $\mathcal{F}^c(\varphi \wedge \psi) = (\mathcal{F}^c(\varphi) \wedge \mathcal{F}^c(\psi))$
4. $\mathcal{F}^c(r!) = \mathcal{R}^c(r)!$
5. $\mathcal{F}^c(r) = \mathcal{R}^c(r)$
6. $\mathcal{F}^c(r \mapsto \varphi) = \mathcal{R}^c(r) \mapsto \mathcal{F}^c(\varphi)$
7. $\mathcal{F}^c(\mathsf{X!}\varphi) = (\neg c\ \mathsf{U}\ (c \wedge \mathsf{X!}\ (\neg c\ \mathsf{U}\ (c \wedge \mathcal{F}^c(\varphi)))))$
8. $\mathcal{F}^c(\varphi\ \mathsf{U}\ \psi) = ((c \rightarrow \mathcal{F}^c(\varphi))\ \mathsf{U}\ (c \wedge \mathcal{F}^c(\psi)))$
9. $\mathcal{F}^c(\varphi\ \mathsf{async\_abort}\ b) = \mathcal{F}^c(\varphi)\ \mathsf{async\_abort}\ b$
10. $\mathcal{F}^c(\varphi\ \mathsf{sync\_abort}\ b) = \mathcal{F}^c(\varphi)\ \mathsf{sync\_abort}\ (b \wedge c)$
11. $\mathcal{F}^c(\mathsf{var}(z)\ \varphi) = \mathsf{var}(z)\ \mathcal{F}^c(\varphi)$
12. $\mathcal{F}^c(\varphi @ c_1) = \mathcal{F}^{c_1}(\varphi)$

Acknowledgments

We would like to thank Shoham Ben-David, John Havlicek, Erich Marschner, Johann Mårtensson, Avigail Orni, Dmitry Pidan and Sitvanit Ruah for many interesting discussions on the subject of local variables. Special thanks to Avigail Orni for important comments on an early version of the LVA semantics.

References

[**B1**] EFHLMV03 - Reasoning with temporal logic over truncated paths

[**B2**] EFHMV03- The definition of a temporal clock operator

[**B5**] EFHM06 - The $\top, \bot$ approach for truncated semantics

[**B6**] EFH05 – A topological characterization of weakness

[**B7**] Fis07 – On the characterization of until as a fixed point under clocked semantics