

Structural Contradictions

Cindy Eisner¹ and Dana Fisman^{1,2}

¹IBM Haifa Research Laboratory ²Hebrew University

Abstract. We study the relation between *logical contradictions* such as $p \wedge \neg p$ and *structural contradictions* such as $p \cap (p \cdot q)$. Intuitively, we expect the two to be treated similarly, but they are not by PSL, nor by SVA. We provide a solution that treats both kinds of contradictions in a consistent manner. The solution reveals that not all structural contradictions are created equal: we must distinguish between them in order to preserve important characteristics of the logic. A happy result of our solution is that it provides the semantics over the natural alphabet 2^P , as opposed to the current semantics of PSL/SVA that use an inflated alphabet including the cryptic letters \top and \perp . We show that the complexity of model checking PSL/SVA is not affected by our proposed semantics.

1 Introduction

A *logical contradiction* is a propositional formula that is not satisfiable, for example $p \wedge \neg p$. In temporal logics such as PSL [8, 17] and SVA [18] that contain semi-extended regular expressions, or SERES, *structural contradictions* arise. A structural contradiction is a SERE that is not satisfiable due to its structure. That is, for any replacement of the propositions in the SERE, the SERE remains unsatisfiable. For example, $p \cap (p \cdot q)$ (the language of words consisting of a single letter on which p holds intersected with the language of words consisting of two letters on which p holds on the first and q on the second) is a structural contradiction, while $(p \cdot q) \cap (p \cdot \neg q)$ is a contradiction, but not a structural one. Structural contradictions were first named in [2], which noted that they were treated differently than logical contradictions by the formal semantics of PSL.

The \cap operator is not the only source of structural contradictions in PSL and SVA. A structural contradiction can also be formed from the fusion operator (denoted here by \circ), a kind of overlapping concatenation. The language of SERE $r_1 \circ r_2$ includes words of the form $v_1 \ell v_2$, where v_1 and v_2 are words, ℓ is a letter, $v_1 \ell$ is in the language of r_1 , and ℓv_2 is in the language of r_2 . The SERE $\lambda \circ p$, where λ denotes the language consisting of the empty word, is a structural contradiction formed without the \cap operator.

Before proceeding, we must first understand a little bit about the use of SERES in PSL/SVA. The set of SERES is built from atoms which are propositional formulas using the standard operators concatenation (which we denote \cdot), union, intersection and Kleene closure, plus the non-standard fusion operator \circ . Let r be a SERE. Then the formula $r!$, a *strong* SERE formula, holds on words containing a non-empty prefix in the language of r . For example:

$$\{a^* \cdot b^* \cdot c\}! \tag{1}$$

holds on a word starting with some number of letters satisfying a followed by some number of letters satisfying b followed by a letter satisfying c , and is equivalent to the LTL formula $[a \text{ U } [b \text{ U } c]]$.

The formula r , a *weak* SERE formula, holds if either $r!$ holds or if the word is “too short” (ends before we have “fallen off” the automaton for r) or “too long” (is an infinite word in which we “got stuck” forever in a starred sub-expression).¹ For example:

$$\{a^* \cdot b^* \cdot c\} \quad (2)$$

holds on words starting with some number of letters satisfying a followed by some number of letters satisfying b followed by a letter satisfying c (in other words, words on which $r!$ holds), on (finite) prefixes of such words (words that are “too short”) and also on infinite words consisting of an infinite number of letters satisfying a or a finite number of letters satisfying a followed by an infinite number of letters satisfying b (words that are “too long”). Recall that Formula 1 is equivalent to $[a \text{ U } [b \text{ U } c]]$. Formula 2, its weak version, is equivalent to the LTL formula $[a \text{ W } [b \text{ W } c]]$.

The problem of structural contradictions arises in the context of weak SERE formulas. Intuitively, a weak SERE formula r is related to its strong counterpart, $r!$, in the same way that the weak until operator **W** is related to its strong counterpart, **U**. For example, the formula $(p^* \cdot q)$ is equivalent to $[p \text{ W } q]$ and the formula $(p^* \cdot q)!$ is equivalent to $[p \text{ U } q]$. However, replacing q with a logical contradiction c gives a different result than replacing it with a structural contradiction s . While $(p^* \cdot c)$ is still equivalent to $[p \text{ W } c]$, the current semantics give that $(p^* \cdot s)$ is not equivalent to $[p \text{ W } s]$, but rather to *false*.

Structural contradictions also arise in the context of formulas using strong operators (e.g., $r!$ and **U**) on *truncated* words, words that are finite but not necessarily maximal. The problem of structural contradictions on such words is created by the PSL **abort** operator and the SVA **disable iff** operator. In [10] it was observed that methods developed for dealing with finite maximal words are not sufficient for dealing with truncated words. On truncated words, the user might want to reason about properties of the truncation as well as properties of the model. For instance, a user might want to specify that a simulation test goes on long enough to discharge all outstanding obligations, or on the other hand, that an obligation need not be met if it “is the fault of the test” (that is, if the test is too short). The former is useful for a test designed to continue until correct output can be confirmed, while the latter approach is useful for a test that “has no opinion” on the correct length of a test – for instance, a monitor checking for bus protocol errors.

The *truncated semantics* of LTL [10] gives three views of the validity of a formula on a truncated word. The views differ with respect to the truth value of a formula if the word was truncated before evaluation of the formula was complete. For example, consider the formula **F** p on a truncated word such that p does not hold for any letter, or the formula **G** q on a truncated word such that q holds at all letters. In both cases we cannot be sure whether or not the formula holds on the original, untruncated word.

A decision to return *true* when there is doubt is termed the *weak view* and a decision to return *false* when there is doubt is termed the *strong view*. Thus in the weak view the formula **F** p holds for any finite word, while **G** q holds only if q holds at every letter of the word. And in the strong view the formula **F** p holds only if p holds at some letter of the word, while the formula **G** q does not hold for any finite word. The *neutral view* demands the maximum that can be reasonably expected from a finite word. Under

¹ Weak SERE formulas are currently part of PSL but not of SVA. However, they are included in the Working Group-approved draft of the next version of SVA.

this approach, the formula $\mathbf{F} p$ holds only if p holds at some letter on the word, while the formula $\mathbf{G} q$ holds only if q holds at every letter on the word. This is exactly the traditional LTL semantics over finite words [19].

Following the truncated semantics, the formula $\mathbf{F} false$ holds in the weak view on a finite word, and the same sort of thing happens on strong SERE formulas in the current semantics of PSL and SVA. Thus the semantics can be understood as considering logical contradictions to be satisfiable in the weak view. The same does not hold, however, for structural contradictions, and the formula $\mathbf{F} (p \cap (p \cdot q))$ does not hold on any word in the weak view, and again, the same sort of thing happens on strong SERE formulas. We provide a solution to this anomaly in the form of a semantics that treats logical and structural contradictions in a consistent manner.

Our solution considers structural contradictions to be satisfiable in the weak view, in the same way that the current semantics of PSL and SVA consider logical contradictions satisfiable in that view. Why do we take this direction, rather than changing instead the treatment of logical contradictions? The reason is that changing the treatment of logical contradictions would have a huge impact on the complexity. In [4] it is shown that the *abort semantics* of an early version of PSL [1] has non-elementary complexity, which can be fixed by the *reset semantics*. These, in turn, were shown by [10] to be equivalent to the *truncated semantics*, which treat logical contradictions as satisfiable in the weak view. The difference in complexity is caused by the decision whether or not to treat logical contradictions as satisfiable in the weak view, thus changing the treatment of logical contradictions would return us to non-elementary complexity.

As we have seen, the idea of considering a logical contradiction to be satisfiable arose independently as a side effect of two very different motivations. While the motivation of the *reset semantics* presented in [4] was complexity, the equivalent *truncated semantics* [10] was motivated by the use of incomplete verification methods. The work presented in this paper shares the second motivation but not the first. Indeed, as we show, the complexity of model checking is not affected by the changes with respect to the existing semantics of PSL and SVA.

Finally, we emphasize that our main motivation is that if logical contradictions must be treated as satisfiable (for whatever reason) then we want logical and structural contradictions to be treated consistently, because intuitively there is no difference between asserting a logical contradiction and asserting a structural one.

2 Preliminaries

PSL and SVA are both temporal logics consisting of LTL [20] operators, formulas formed from semi-extended regular expressions (SEREs), clock and abort operators, and a lot of syntactic sugar.² It is of course not necessary to consider the syntactic sugar. Furthermore, clocks are orthogonal to the issues we examine in this paper (and as shown by [12], can be written away by the rewrite rules of [17, 18]), thus we use as our base logic the logic PSC (for PSL/SVA Core), consisting of all of the above-mentioned operators except for the clock operator. Since in the absence of the clock operator there is

² While the current version of SVA does not have LTL operators, they are included in the Working Group-approved draft of the next version of SVA.

Definition 1 (Semi-extended Regular Expressions (SERES)) If $b \in B$ is a propositional formula and $r, r_1,$ and r_2 are SERES, then the following are SERES:						
$\bullet \lambda$	$\bullet b$	$\bullet r_1 \cdot r_2$	$\bullet r_1 \circ r_2$	$\bullet r^*$	$\bullet r_1 \cup r_2$	$\bullet r_1 \cap r_2$
Definition 2 (PSC formulas) If $b \in B$ is a propositional formula, φ and ψ are PSC formulas and r a SERE, then the following are PSC formulas:						
$\bullet \neg\varphi$	$\bullet \varphi \wedge \psi$	$\bullet X!\varphi$	$\bullet [\varphi \text{ U } \psi]$			
$\bullet \varphi \text{ abort } b$	$\bullet r!$	$\bullet r$	$\bullet r \mapsto \varphi$			

Fig. 1: The syntax of PSC

no difference between synchronous and asynchronous abort, we use here a single abort operator which we denote **abort**. We are left with LTL operators plus formulas formed from semi-extended regular expressions (SERES) and the abort operator.

We define PSC formulas with respect to a non-empty set of atomic propositions P and a given set of propositional formulas B over P . Then SERES and PSC formulas are defined as shown in Definitions 1 and 2 in Figure 1. Note that every propositional formula is a SERE, thus the weak and strong Booleans of [17] are included.

The weak next operator, X , is needed in order to deal with finite words [19] and is defined as syntactic sugaring as follows: $X\varphi = \neg X!\neg\varphi$. In this paper we also make use of $F\varphi = [\text{true} \text{ U } \varphi]$ and $G\varphi = \neg F\neg\varphi$.

LTL is the subset of PSC consisting of degenerate SERES of the form b as the base case, the Boolean connectives \neg and \wedge , and the temporal operators $X!$ and U .

2.1 Notation

Let $\Sigma = 2^P \cup \{\top, \perp\}$; thus a letter is either a subset of the set of atomic propositions P or one of the special letters \top, \perp . We will denote a letter from Σ by ℓ and an empty, finite, or infinite word from Σ by u, v or w . We denote the length of word w as $|w|$. An empty word $w = \epsilon$ has length 0, a finite word $w = (\ell_0\ell_1\ell_2 \dots \ell_n)$ has length $n + 1$, and an infinite word has length ∞ . We denote the i^{th} letter of w by w^{i-1} (since counting of letters starts at zero). We denote by $w^{i\cdot\cdot}$ the suffix of w starting at w^i . That is, $w^{i\cdot\cdot} = (w^i w^{i+1} \dots w^n)$ or $w^{i\cdot\cdot} = (w^i w^{i+1} \dots)$. We denote by $w^{i\cdot\cdot j}$ the finite sequence of letters starting from w^i and ending in w^j . That is, $w^{i\cdot\cdot j} = (w^i w^{i+1} \dots w^j)$. We make use of an ‘‘overflow’’ and ‘‘underflow’’ for the indices of w . That is, $w^{j\cdot\cdot} = \epsilon$ if $j \geq |w|$, and $w^{j\cdot\cdot k} = \epsilon$ if $j \geq |w|$ or $k < j$.

The dual word of w , denoted \bar{w} , is the word obtained by replacing every \top with a \perp and vice versa. We use *true* and *false* to denote $p \vee \neg p$ and $p \wedge \neg p$, respectively, for some atomic proposition p . We use φ and ψ to denote PSC formulas, p to denote an atomic proposition, and j and k to denote natural numbers.

For languages L_1 and L_2 we use $L_1 \cdot L_2$ to denote the set $\{w_1 w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$ and $L_1 \circ L_2$ to denote the set $\{w_1 \ell w_2 \mid w_1 \ell \in L_1 \text{ and } \ell w_2 \in L_2\}$. For a language L we use L^0 to denote $\{\epsilon\}$, L^i to denote $L^{i-1} \cdot L$, L^* to denote $\bigcup_{i \geq 0} L^i$ and L^+ to de-

Definition 3 (The language of SERE r)	
•	$\mathcal{L}(\lambda) = \epsilon$
•	$\mathcal{L}(b) = \{\ell \mid \ell \models b\}$
•	$\mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)$
•	$\mathcal{L}(r_1 \circ r_2) = \mathcal{L}(r_1) \circ \mathcal{L}(r_2)$
•	$\mathcal{L}(r^*) = \mathcal{L}(r)^*$
•	$\mathcal{L}(r_1 \cup r_2) = \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$
•	$\mathcal{L}(r_1 \cap r_2) = \mathcal{L}(r_1) \cap \mathcal{L}(r_2)$

Fig. 2: The language of SERE r

note $\bigcup_{i>0} L^i$. We use L^ω for the language composed of infinitely many concatenations of L with itself. For an infinite word w , we define that $wv = w$ for any word v .

We use $\langle a \rangle$ to denote a letter on which atomic proposition a and only atomic proposition a holds, $\langle ab \rangle$ to denote a letter on which atomic propositions a and b and only atomic propositions a and b hold, etc. Thus $\langle a \rangle \langle bc \rangle \langle d \rangle$ describes a finite word of three letters, such that a is the only atomic proposition that holds on the first letter, b and c are the only atomic propositions that hold on the second letter, and d is the only atomic proposition that holds on the third letter.

2.2 The current semantics of PSC

The current semantics of PSC in [17, 18] is defined inductively, using as the base case the semantics of propositional formulas over letters in $\Sigma = 2^P \cup \{\top, \perp\}$. The semantics of propositional formulas is assumed to be given as a relation $\models \subseteq \Sigma \times B$ relating letters in Σ with propositional formulas in B . If $(\ell, b) \in \models$ we say that the letter ℓ satisfies the propositional formula b and denote it $\ell \models b$. We assume that the two special letters \top and \perp behave as follows: for every propositional formula b , $\top \models b$ and $\perp \not\models b$. We further assume that on every other letter, $\ell \models p$ iff $p \in \ell$. Finally, we assume that otherwise the relation \models behaves in the usual manner, and in particular that Boolean disjunction, conjunction and negation behave as usual.

The language of SERE r (the words on which r holds tightly [17], or is tightly satisfied [18]), denoted $\mathcal{L}(r)$, is defined formally as shown in Definition 3 in Figure 2.

Because of its use of the special letters \top and \perp , we call the semantics of [17, 18] the \top, \perp approach to the semantics of PSC, and we use $w \models_{\top, \perp} \varphi$ to denote that w models φ under the \top, \perp approach. The semantics are given in Definition 4 in Figure 3. The semantics of the LTL operators are standard, except that we use the dual word when negating. The semantics of $r!$ are straightforward: $r!$ holds on a word w if there exists a finite prefix of w that is in $\mathcal{L}(r)$. For example, $(a \cdot b \cdot c)!$ holds on the word $u = \langle a \rangle \langle b \rangle \langle c \rangle \langle d \rangle$.

Intuitively, the semantics of r are that nothing should go wrong before reaching a final state in the automaton of r . In other words, that if we have not completed a word in $\mathcal{L}(r)$, then at least things have not gone so wrong that appending some number of

Definition 4 (The current semantics of PSC)

- $w \models_{\perp} \neg\varphi \iff \bar{w} \not\models_{\perp} \varphi$
 - $w \models_{\perp} \varphi \wedge \psi \iff w \models_{\perp} \varphi \text{ and } w \models_{\perp} \psi$
 - $w \models_{\perp} X!\varphi \iff |w| > 1 \text{ and } w^{1..} \models_{\perp} \varphi$
 - $w \models_{\perp} [\varphi \cup \psi] \iff \exists k < |w| \text{ such that } w^{k..} \models_{\perp} \psi \text{ and for every } j < k, w^{j..} \not\models_{\perp} \varphi$
-
- $w \models_{\perp} r! \iff \exists j < |w| \text{ s.t. } w^{0..j} \in \mathcal{L}(r)$
 - $w \models_{\perp} r \iff \forall j < |w|, w^{0..j} \top^{\omega} \models_{\perp} r!$
 - $w \models_{\perp} r \mapsto \varphi \iff \forall j < |w| \text{ s.t. } \bar{w}^{0..j} \in \mathcal{L}(r), w^{j..} \models_{\perp} \varphi$
-
- $w \models_{\perp} \varphi \text{ abort } b \iff \text{either } w \models_{\perp} \varphi \text{ or } \exists j < |w| \text{ s.t. } w^j \models b \text{ and } w^{0..j-1} \top^{\omega} \models_{\perp} \varphi$

Fig. 3: The current semantics of PSC (the \top, \perp approach)

\top 's won't get us there. Thus the semantics are that for every prefix $w^{0..j}$ of w , $w^{0..j} \top^{\omega}$ should satisfy $r!$. For example, $(a \cdot b^* \cdot c)$ holds on the word $v_1 = \langle a \rangle \langle b \rangle \langle c \rangle \langle d \rangle$, and also on the “too short” word $v_2 = \langle a \rangle \langle b \rangle$ and on the “too long” word $v_3 = \langle a \rangle \langle b \rangle^{\omega}$.

The semantics of $r \mapsto \varphi$ (read “ r suffix implies φ ”) are that whenever we see a prefix of w in $\mathcal{L}(r)$, we must have that φ holds on the continuation of w starting at the last letter of the prefix that “matched” r .³ For example, the formula $(a \cdot b \cdot c) \mapsto \mathbf{G} d$ holds on the word $\langle a \rangle \langle b \rangle \langle cd \rangle \langle d \rangle^{\omega}$ because $\mathbf{G} d$ holds on the suffix of the word starting from the last letter that “matched” $(a \cdot b \cdot c)$. Note the use of the dual word \bar{w} . This is because implication entails a negation of the left operand. For example, the formula $(a \cdot b \cdot c) \mapsto \varphi$ does not hold on the word $w = \langle a \rangle \langle b \rangle \perp^{\omega}$ for any φ , because we have that $\bar{w} = \langle a \rangle \langle b \rangle \top^{\omega}$, and letting $j = 3$ we find the word $\langle a \rangle \langle b \rangle \top \in \mathcal{L}(a \cdot b \cdot c)$, but the continuation of the original word w starting from the third letter is \perp^{ω} , on which φ surely does not hold (recall that nothing holds on \perp , not even *true*).

Finally, the semantics of the abort operator are that we truncate the word at the point where we see the abort condition, and pad the result with an infinite number of \top 's. Intuitively, this has the effect of weakening the strong operators on the truncated word. To see this, let the original word w be $w = uv$, and let the truncated word be u . Then a condition that must be fulfilled due to a strong operator can be fulfilled by one of the \top 's, if it is not fulfilled on u . For example, consider formula $\varphi = [p \cup q] \text{ abort } b$ on word $w = \langle p \rangle \langle p \rangle \langle p \rangle \langle b \rangle$. Truncating w at b gives us $u = \langle p \rangle \langle p \rangle \langle p \rangle$. Then q holds on every \top and in particular on the first one, so $[p \cup q]$ holds on $u \top^{\omega}$.

3 Structural contradictions

The \top, \perp approach, used by both PSL [17] and SVA [18], is broken with respect to structural contradictions. We first define formally what we mean by a logical and a structural contradiction, then show the problem and our solution.

³ In Dynamic Logic [13, 15], $r \mapsto \varphi$ corresponds to $[r]\varphi$.

Definition 5 (Logical contradiction) A logical contradiction is a propositional formula that is not satisfiable.

Observation 1 A logical contradiction can be transformed into a satisfiable (and valid) formula by substituting each proposition with either true or false.

Definition 6 (Structural contradiction) A structural contradiction is a SERE that is not satisfiable, and that remains unsatisfiable under every substitution of each proposition with either true or false.

3.1 The problem

Let $\varphi = (a \cdot b^* \cdot \text{false})$ and let $\varphi' = (a \cdot b^* \cdot (c \cap (c \cdot c)))$. It is easy to see that φ holds on $w = \langle a \rangle \langle b \rangle \langle b \rangle \langle b \rangle$ but that φ' does not. The reason for this is that any propositional formula holds on the letter \top , thus adding \top 's to the end of a finite word allows us to satisfy any logical contradiction in the future. However, there is no word, not even one consisting entirely of \top 's, on which a structural contradiction holds.

As suggested in [9], it seems that allowing the special letter \top to match not only any propositional formula, but also any SERE of any length, might make the problem disappear by definition. Under the *flexible letter* approach, we modify the \top, \perp approach by defining that $\mathcal{L}(\lambda) = \top^*$ and that $\mathcal{L}(b) = \{\ell \mid \ell \models b\} \cdot \top^*$. We then get that $\varphi' = (a \cdot b^* \cdot (c \cap (c \cdot c)))$ holds on $w = \langle a \rangle \langle b \rangle \langle b \rangle \langle b \rangle$ as we want. To see this, note that for φ to hold on w we need that $w \top^\omega \models_{\perp} \varphi$. We have that $\top \top \in \mathcal{L}(c \cap (c \cdot c))$ because $\top \top \in \mathcal{L}(c)$ and also that $\top \top \in \mathcal{L}(c \cdot c)$. Thus we can let $j = 5$ and take the first six letters of $w \top^\omega$ to be in $\mathcal{L}(a \cdot b^* \cdot (c \cap (c \cdot c)))$.

The problem with this solution is that it has a harmful side effect – it changes the semantics of formulas that do not contain structural contradictions. Let $r_1 = (a \cdot b \cdot c)$, let $r_2 = (d \cdot e \cdot f)$, let $\varphi = (r_1 \circ r_2)$ and let $w = \langle a \rangle \langle b \rangle \langle c \rangle$. Then under the \top, \perp approach, using the original semantics of $\mathcal{L}(r)$ as defined in Section 2.2, we have that φ does not hold on w . To see this, note that $|w| = 3$. Thus $w \models_{\perp} \varphi$ iff $\forall j < 3$ we have that $w^{0..j} \top^\omega \models_{\perp} (r_1 \circ r_2)!$. Taking $j = 2$ we get that $\langle a \rangle \langle b \rangle \langle c \rangle \top^\omega \not\models_{\perp} (r_1 \circ r_2)!$, because that requires one letter overlap between $(a \cdot b \cdot c)$ and $(d \cdot e \cdot f)$, but we do not have that d holds on the third letter. Under the flexible letter approach, using the modified definitions of $\mathcal{L}(\lambda)$ and $\mathcal{L}(p)$ of the previous paragraph, we can use more than three letters to “match” $(a \cdot b \cdot c)$. So take the first four letters of $\langle a \rangle \langle b \rangle \langle c \rangle \top^\omega$ to be in $\mathcal{L}(a \cdot b \cdot c)$, and then, starting from the last letter of those four (which is \top), take some number of letters to be in $\mathcal{L}(d \cdot e \cdot f)$. Since all letters after the first \top are \top , we can take three \top 's for that. Then the overlap happens on a \top , and thus under the flexible letter approach we get that φ holds on w .

3.2 The solution

Our proposed semantics for PSC is based on the *truncated semantics* of LTL [10]. The truncated semantics is defined over the natural alphabet 2^P and gives three views of the truth value of a formula, roughly corresponding to the truth value on $w \top^\omega$ (the weak view), w (the neutral view) and $w \perp^\omega$ (the strong view). Formally, the truncated

Definition 7 (Truncated semantics of LTL [10])

holds weakly: For w over 2^P such that $|w| \geq 0$,

- $w \models^- b \iff |w| = 0 \text{ or } w^0 \models b$
- $w \models^- \neg\varphi \iff w \not\models^+ \varphi$
- $w \models^- \varphi \wedge \psi \iff w \models^- \varphi \text{ and } w \models^- \psi$
- $w \models^- X! \varphi \iff w^{1..} \models^- \varphi$
- $w \models^- [\varphi \text{ U } \psi] \iff \exists k \text{ such that } w^{k..} \models^- \psi \text{ and for every } j < k, w^{j..} \models^- \varphi$
- $w \models^- \varphi \text{ abort } b \iff \text{either } w \models^- \varphi \text{ or } \exists j < |w| \text{ s.t. } w^j \models b \text{ and } w^{0..j-1} \models^- \varphi$

holds neutrally: For w over 2^P such that $|w| > 0$,

- $w \models b \iff w^0 \models b$
- $w \models \neg\varphi \iff w \not\models \varphi$
- $w \models \varphi \wedge \psi \iff w \models \varphi \text{ and } w \models \psi$
- $w \models X! \varphi \iff |w| > 1 \text{ and } w^{1..} \models \varphi$
- $w \models [\varphi \text{ U } \psi] \iff \exists k < |w| \text{ such that } w^{k..} \models \psi \text{ and for every } j < k, w^{j..} \models \varphi$
- $w \models \varphi \text{ abort } b \iff \text{either } w \models \varphi \text{ or } \exists j < |w| \text{ s.t. } w^j \models b \text{ and } w^{0..j-1} \models^- \varphi$

holds strongly: For w over 2^P such that $|w| \geq 0$,

- $w \models^+ b \iff |w| > 0 \text{ and } w^0 \models b$
- $w \models^+ \neg\varphi \iff w \not\models^- \varphi$
- $w \models^+ \varphi \wedge \psi \iff w \models^+ \varphi \text{ and } w \models^+ \psi$
- $w \models^+ X! \varphi \iff w^{1..} \models^+ \varphi$
- $w \models^+ [\varphi \text{ U } \psi] \iff \exists k \text{ such that } w^{k..} \models^+ \psi \text{ and for every } j < k, w^{j..} \models^+ \varphi$
- $w \models^+ \varphi \text{ abort } b \iff \text{either } w \models^+ \varphi \text{ or } \exists j < |w| \text{ s.t. } w^j \models b \text{ and } w^{0..j-1} \models^- \varphi$

Fig. 4: The truncated semantics of LTL [10]

semantics of LTL is defined as shown in Definition 7 in Figure 4, where **abort** is the truncation operator, termed `trunc.w` by [10].

The neutral view is defined over non-empty words and corresponds to the traditional LTL semantics over finite words, while the weak and strong views are defined on empty words as well. The weak/strong views consider the empty word explicitly in the semantics of b (the weak view considers it sufficient that $|w| = 0$ while the strong view requires that w be non-empty), while the neutral view assumes that w is non-empty.

A third difference is that the weak and strong views make use of the definition of “overflow” for the indices of w . For example, in Definition 7, consider the semantics of $[\varphi \text{ U } \psi]$ under the weak and strong views. When we say “ $\exists k$ ”, k is not required to be less than $|w|$. In the weak view, this has the effect of allowing all eventualities to hold, because an overflow results in ϵ , b holds on ϵ , and the rest follows easily by induction.

Definition 8 (The weak language of SERE r)	
•	$\mathcal{L}_{weak}(\lambda) = \epsilon$
•	$\mathcal{L}_{weak}(b) = \epsilon \cup \{\ell \mid \ell \models b\}$
•	$\mathcal{L}_{weak}(r_1 \cdot r_2) = \mathcal{L}_{weak}(r_1) \cup (\mathcal{L}(r_1) \cdot \mathcal{L}_{weak}(r_2))$
•	$\mathcal{L}_{weak}(r_1 \circ r_2) = \mathcal{L}_{weak}(r_1) \cup (\mathcal{L}(r_1) \circ \mathcal{L}_{weak}(r_2))$
•	$\mathcal{L}_{weak}(r^*) = (\mathcal{L}(r)^* \cdot \mathcal{L}_{weak}(r)) \cup \mathcal{L}(r)^\omega$
•	$\mathcal{L}_{weak}(r_1 \cup r_2) = \mathcal{L}_{weak}(r_1) \cup \mathcal{L}_{weak}(r_2)$
•	$\mathcal{L}_{weak}(r_1 \cap r_2) = \mathcal{L}_{weak}(r_1) \cap \mathcal{L}_{weak}(r_2)$

Fig. 5: The weak language of SERE r

In the strong view, the overflow has the effect of not allowing any eventuality to hold, by similar reasoning.

The view is preserved in the inductive definition (e.g., \wedge uses \models in the weak view, \models in the neutral view and \models^\pm in the strong view) except that negation switches between the views, and aborting a formula always takes us to the weak view. Altogether, we get that the formula φ holds on a truncated word in the weak view if up to the end of the word, “nothing has yet gone wrong” with φ . It holds on a truncated word in the neutral view according to the standard LTL semantics on finite words. It holds on a truncated word in the strong view if everything that needs to happen in order to convince us that φ holds on the original, untruncated word, has already occurred.

For LTL formulas, the truncated semantics are equivalent to the \top, \perp approach, as shown in [11].

Theorem 2 ([11]) *Let u be a word over 2^P , let v be a non-empty word over 2^P , and let φ be an LTL formula. Then, as shown in [11], the following equivalences hold:*

$$\bullet u \models^- \varphi \iff u \top^\omega \models_{\perp} \varphi \quad \bullet v \models \varphi \iff v \models_{\perp} \varphi \quad \bullet u \models^+ \varphi \iff u \perp^\omega \models_{\perp} \varphi$$

On infinite words, the weak, neutral and strong views coincide, as shown by [10]. Note that this follows trivially from the \top, \perp approach, by the definition of concatenation to the right of an infinite word u .

We are now ready to extend the truncated semantics to all of PSC. As a first step, we can try to extend the definition of the *weak language* of a SERE [9] to the intersection and fusion operators. As defined in [9], the weak language of a SERE includes words that are in $\mathcal{L}(r)$, words that are “too short” (are finite prefixes of words in $\mathcal{L}(r)$), and also words that are “too long” (in which we get stuck in a starred sub-expression). The weak language of a SERE is defined in Definition 8 in Figure 5.

Using the weak language $\mathcal{L}_{weak}(r)$, we can define that

$$w \models r \iff \text{either } \exists j < |w| \text{ s.t. } w^{0..j} \in \mathcal{L}(r) \text{ or } w \in \mathcal{L}_{weak}(r)$$

This appears to give us what we want, but there are two complications. The first is that we get that the formula $\varphi = (a \cdot b \cdot c) \circ (d \cdot e \cdot f)$ holds on the word $w = \langle a \rangle \langle b \rangle \langle c \rangle$. Thus

<p>Definition 9 (The language of finite proper prefixes of SERE r) The language of finite proper prefixes of SERE r, denoted $\mathcal{F}(r)$, is defined as follows:</p> <ul style="list-style-type: none"> • $\mathcal{F}(\lambda) = \emptyset$ • $\mathcal{F}(b) = \epsilon$ <ul style="list-style-type: none"> • $\mathcal{F}(r_1 \cdot r_2) = \mathcal{F}(r_1) \cup (\mathcal{L}(r_1) \cdot \mathcal{F}(r_2))$ • $\mathcal{F}(r_1 \circ r_2) = \mathcal{F}(r_1) \cup (\mathcal{L}(r_1) \circ \mathcal{F}(r_2))$ • $\mathcal{F}(r^*) = \mathcal{L}(r)^* \cdot \mathcal{F}(r)$ <ul style="list-style-type: none"> • $\mathcal{F}(r_1 \cap r_2) = \mathcal{F}(r_1) \cap \mathcal{F}(r_2)$ • $\mathcal{F}(r_1 \cup r_2) = \mathcal{F}(r_1) \cup \mathcal{F}(r_2)$ 	<p>Definition 10 (The loop language of SERE r) The loop language of SERE r, denoted $\mathcal{I}(r)$, is defined as follows:</p> <ul style="list-style-type: none"> • $\mathcal{I}(\lambda) = \emptyset$ • $\mathcal{I}(b) = \emptyset$ <ul style="list-style-type: none"> • $\mathcal{I}(r_1 \cdot r_2) = \mathcal{I}(r_1) \cup (\mathcal{L}(r_1) \cdot \mathcal{I}(r_2))$ • $\mathcal{I}(r_1 \circ r_2) = \mathcal{I}(r_1) \cup (\mathcal{L}(r_1) \circ \mathcal{I}(r_2))$ • $\mathcal{I}(r^*) = (\mathcal{L}(r)^* \cdot \mathcal{I}(r)) \cup \mathcal{L}(r)^\omega$ <ul style="list-style-type: none"> • $\mathcal{I}(r_1 \cap r_2) = \mathcal{I}(r_1) \cap \mathcal{I}(r_2)$ • $\mathcal{I}(r_1 \cup r_2) = \mathcal{I}(r_1) \cup \mathcal{I}(r_2)$
--	---

Fig. 6: The languages $\mathcal{F}(r)$ and $\mathcal{I}(r)$

we have changed the semantics with respect to formulas not containing structural contradictions, exactly the problem that we had with the flexible letter approach. To see this, let $r_1 = (a \cdot b \cdot c)$, let $r_2 = (d \cdot e \cdot f)$, and examine the semantics of $\mathcal{L}_{weak}(r_1 \circ r_2)$. Since we have already seen the third letter, we want to insist that d holds on it, as we do in the current semantics of PSC. But since $w \in \mathcal{L}_{weak}(r_1)$, we have not done so.

The second complication is that $\mathcal{L}_{weak}(r)$ does not distinguish between the case where formula r holds because the word is “too short” (is finite) and the case where it holds because the word is “too long” (is infinite). Thus it will be difficult for us to use $\mathcal{L}_{weak}(r)$ to extend the truncated semantics, in which it is required that the three views coincide on infinite, but not on finite words.

We address the first issue by defining a language that stops one letter short of words in $\mathcal{L}(r)$. In order to address the second issue, we split $\mathcal{L}_{weak}(r)$ into two languages. The language of proper prefixes of a SERE, denoted $\mathcal{F}(r)$, consists of *finite* proper prefixes of words in $\mathcal{L}(r)$, except that logical and structural contradictions are considered satisfiable. The loop language of a SERE, denoted $\mathcal{I}(r)$, extends the *loop*(\cdot) of [16] to the intersection and fusion operators. It consists of *infinite* words in which we get “stuck forever” in a starred sub-expression of r . Formally, these languages are defined as in Definitions 9 and 10 in Figure 6.

Using $\mathcal{L}(r)$, $\mathcal{F}(r)$ and $\mathcal{I}(r)$, we extend the truncated semantics to all of PSC as shown in Definition 11 in Figure 7. Note that whenever we ask whether $w \in \mathcal{F}(r)$ we also ask if $w \in \{\epsilon\}$. This is because $\epsilon \notin \mathcal{F}(\lambda)$ (it recognizes only proper prefixes), however we want to preserve the property of [10] that any formula holds weakly on ϵ .

Examine now the semantics of $r!$ under the neutral view. The formula $r!$ holds neutrally on w if there exists a non-empty prefix of w that is in $\mathcal{L}(r)$. Since formula $r!$ is strong, its semantics under the strong view are identical to its semantics under the neutral view. Under the weak view, the semantics are weakened. The weakening includes the empty word and words in $\mathcal{F}(r)$, but not words in $\mathcal{I}(r)$. This is because the views should differ only on truncated words, which are finite, while words in the loop language are infinite.

Definition 11 (Truncated semantics of PSC) *The truncated semantics of PSC consists of the truncated semantics of LTL as shown in Definition 7 in Figure 4, extended to the SERE-based operators as follows:*

holds weakly: For w over 2^P such that $|w| \geq 0$,

- $w \models^- r!$ \iff either $\exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ or $w \in \mathcal{F}(r) \cup \{\epsilon\}$
- $w \models^- r$ \iff either $\exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ or $w \in \mathcal{I}(r)$ or $w \in \mathcal{F}(r) \cup \{\epsilon\}$
- $w \models^- r \mapsto \varphi$ $\iff \forall j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ we have $w^{j..} \models^- \varphi$

holds neutrally: For w over 2^P such that $|w| > 0$,

- $w \models r!$ $\iff \exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$
- $w \models r$ \iff either $\exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ or $w \in \mathcal{I}(r)$ or $w \in \mathcal{F}(r) \cup \{\epsilon\}$
- $w \models r \mapsto \varphi$ $\iff \forall j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ we have $w^{j..} \models \varphi$

holds strongly: For w over 2^P such that $|w| \geq 0$,

- $w \models^+ r!$ $\iff \exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$
- $w \models^+ r$ $\iff \exists j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ or $w \in \mathcal{I}(r)$
- $w \models^+ r \mapsto \varphi$ $\iff w \notin \mathcal{F}(r) \cup \{\epsilon\}$ and $\forall j < |w|$ s.t. $w^{0..j} \in \mathcal{L}(r)$ we have $w^{j..} \models^+ \varphi$

Fig. 7: Truncated semantics of PSC

The formula r holds neutrally on w if either there exists a non-empty prefix of w that is in $\mathcal{L}(r)$, or w is “too long” (is in $\mathcal{I}(r)$) or “too short” (is in $\mathcal{F}(r) \cup \{\epsilon\}$). Since formula r is weak, its semantics under the weak view are identical to its semantics under the neutral view. Under the strong view, the semantics are strengthened. Similarly to the weakening of $r!$ in the weak view, the strengthening affects only finite words.

The formula $r \mapsto \varphi$ holds neutrally on w if for every finite non-empty prefix of w in $\mathcal{L}(r)$, φ holds on the suffix of w starting from the last letter of the prefix. Formula $r \mapsto \varphi$ is weak, therefore its semantics under the weak view are similar to its semantics under the neutral view (the difference is in the strength used to check φ). Under the strong view, the semantics are strengthened. The strengthening involves eliminating ϵ and words that are in $\mathcal{F}(r)$. To understand this, notice that on such words there might exist an extension to a word in $\mathcal{L}(r)$ on the original, untruncated word (considering logical and structural contradictions to be satisfiable) and we cannot be sure whether φ holds on the original, untruncated word starting from that point.

Before examining the characteristics of the truncated semantics of PSC, let’s do a quick sanity check. Previously we saw that on a finite word $w = \langle a \rangle \langle b \rangle \langle b \rangle \langle b \rangle$, formulas $\varphi = (a \cdot b^* \cdot \text{false})$ and $\varphi' = (a \cdot b^* \cdot (c \cap (c \cdot c)))$ behave differently under the \top, \perp approach. Under the extension to the truncated semantics, we get that both of them hold on w . Thus it seems that in some sense the truncated semantics of PSC is treating logical and structural contradictions in a consistent manner. In the next section we formalize this and other characteristics of our solution.

4 Characteristics of the truncated semantics of PSC

In this section we show that our extension of the truncated semantics to all of PSC treats logical and structural contradictions in a consistent manner. We show that it preserves important properties of the original truncated semantics, and that for formulas without structural contradictions, it preserves the semantics of the \top, \perp approach. Finally, we show that the complexity of model checking is the same under the truncated semantics of PSC as under the \top, \perp approach. We start with a simple proposition.⁴

Proposition 3 (*false vs. true* \cap (*true* \cdot *true*)) *Let φ be a PSC formula containing false and let φ' be the formula obtained by replacing every occurrence of false with the structural contradiction $true \cap (true \cdot true)$. Let w be a finite or infinite word over 2^P . Then the truth values of φ and φ' on w agree under the truncated semantics of PSC.*

We would like to extend Proposition 3 to cover any logical vs. structural contradiction, and not just *false vs. true* \cap (*true* \cdot *true*). However, things are not so simple. While every logical contradiction is equivalent to every other logical contradiction, not all structural contradictions are created equal. For example, let $r_1 = true \cap (true \cdot true)$ and let $r_2 = (true \cdot true) \cap (true \cdot true \cdot true)$. Then r_2 holds on words of length 1, while r_1 does not. To understand this, note that a structural contradiction contains an element of temporality that is not present in a logical contradiction. So we should compare a structural contradiction to a temporal logic formula that also contains such an element. For example, while r_1 is equivalent to *false*, r_2 is equivalent to $true \cdot (true \cap (true \cdot true))$, which is equivalent to $X false$.

We define the *order* of a formula to be the length of the longest word on which it holds weakly. Thus *false* and $p \cap (p \cdot q)$ are of order 0 and $X false$ and $(p \cdot q) \cap (p \cdot q \cdot r)$ are of order 1, and we expect that a generalization of Proposition 3 would compare only formulas of the same order. Note first that two formulas of order n are not necessarily equivalent. For example, $X false$ holds on any word of length 1, whereas $(p \cdot q) \cap (p \cdot q \cdot r)$ holds only on a subset of such words – those where p holds on the first letter.

Let $X^0 \varphi$ denote φ , let X^n denote n repetitions of the X operator and let r^n denote n concatenations of r with itself. Then Proposition 4 below generalizes Proposition 3.

Proposition 4 (X^{n-1} *false vs. true* ^{n} \cap (*true* ^{n} \cdot *true* ^{$+$})) *Let $n \geq 1$, let φ be a PSC formula containing sub-formula $\psi = X^{n-1} false$ and let φ' be the formula obtained by replacing every occurrence of ψ with the structural contradiction $\psi' = true^n \cap (true^n \cdot true^+)$. Let w be a finite or infinite word over 2^P . Then the truth values of φ and φ' on w agree under the truncated semantics of PSC.*

The observant reader may have noticed that the structural contradiction $\lambda \cap true$ is not addressed by Proposition 4. It has order 0, and it might be expected that $true \cap (true \cdot true) = true \cdot (\lambda \cap true)$ have order 1 (it actually has order 0). This apparent anomaly is resolved if we consider a clocked semantics [12]. Under a clocked semantics, $X^0 \varphi$ is not equivalent to φ , but is the *clock alignment operator*, and we have that

⁴ All proofs appear in the full version of this paper, available at <http://www.cs.huji.ac.il/danafi/publications>.

$\lambda \cap true$ is equivalent to *false* while $true \cap (true \cdot true)$ is equivalent to X^0 *false*. The details are beyond the scope of this paper – for a thorough discussion of clock alignment, see [14]. Finally, note that all structural contradictions of the form $\lambda \circ r$ are of order 0, and are equivalent to *false*.

Proposition 4 deals with structural contradictions of a specific structure. What about others, for example $\lambda \cap true$, $(true^{13} \cap true^7) \cup (true^9 \cap true^4)$, $(a^6 \cup a^4) \cap a^7$, or $(a \cdot b) \cap (a \cdot b \cdot c \cdot d \cdot e)$? Are they satisfiable in the weak view? The answer is yes, all structural contradictions are satisfied on the empty word under the weak view (and none is satisfied by the empty word in the strong view), as stated in the following lemma.

Lemma 5 *Let φ be a formula in PSC. Then both $\epsilon \models^- \varphi$ and $\epsilon \not\models^\pm \varphi$.*

Lemma 5 fulfills our intuition that on the empty word clearly nothing has yet gone wrong (thus it should accept weakly any formula, including a structural contradiction) and conversely the empty word provides evidence for nothing (thus it should accept no word strongly, including a structural contradiction). Lemma 5 is important because it shows that a fundamental property of the original truncated semantics, one that was broken in the \top, \perp approach, is preserved by the truncated semantics of PSC. Below we show that other important properties of the original truncated semantics are preserved by our extension to all of PSC. Theorem 6 states that the strong view is stronger than the neutral, which is in turn stronger than the weak.

Theorem 6 (Strength relation theorem) *Let w be a non-empty word over 2^P , and φ be a formula in PSC. Then*

$$\bullet w \models^\pm \varphi \implies w \models \varphi \qquad \bullet w \models \varphi \implies w \models^- \varphi$$

Corollary 7 states that all three views are equivalent over infinite words.

Corollary 7 *If w is infinite, then $w \models^- \varphi$ iff $w \models \varphi$ iff $w \models^\pm \varphi$.*

Finally, Theorem 8 states that if a formula holds weakly on a word w then it holds weakly on all prefixes of w , and if a formula holds strongly on a word w then it holds strongly on all extensions of w . We say that u is a prefix of v , denoted $u \preceq v$, if there exists a word u' such that $uu' = v$. We say that w is an extension of v , denoted $w \succeq v$, if there exists a word v' such that $w = vv'$.

Theorem 8 (Prefix/extension theorem) *Let v be a word over 2^P , and φ be a formula in PSC. Then*

$$\bullet v \models^- \varphi \iff \forall u \preceq v, u \models^- \varphi \qquad \bullet v \models^\pm \varphi \iff \forall w \succeq v, w \models^\pm \varphi$$

Previously we have seen that even without structural contradictions, the flexible letter approach and the solution using $\mathcal{L}_{weak}(r)$ differ from the semantics of the \top, \perp approach. The theorem below states that without structural contradictions, the \top, \perp approach and our extension to the truncated semantics agree.

Theorem 9 *Let u be a word over 2^P , v be a non-empty word over 2^P , and φ be a PSC formula that does not contain a SERE that is a structural contradiction. Then*

$$\bullet u \models^- \varphi \iff u \top^\omega \models_{\perp}^- \varphi \qquad \bullet v \models \varphi \iff v \models_{\perp} \varphi \qquad \bullet u \models^\pm \varphi \iff u \perp^\omega \models_{\perp}^\pm \varphi$$

Theorem 9 shows that the special letters \top and \perp are tools that are useful in stating the \top, \perp approach, but are not necessary in stating the semantics in the absence of structural contradictions. Besides the fact that they don't give us the semantics that we want, a major disadvantage of the \top, \perp approach is that it uses the special letters \top and \perp to encrypt the semantics, rather than stating directly what they are, as we do in the extension to the truncated semantics.

Intuitively, the weak view can be obtained by considering all states of a Büchi automaton to be accepting, similarly to the *safety component* of [3] and to the *looping acceptance condition* of [21]. However, we have to be careful. The particular automaton that results from applying the looping acceptance condition depends on the form of the automaton that we start with, and not just on its original language. For instance, consider the Büchi automaton consisting of a single (not final) state and no transitions. Obviously, its language is empty. Now add a single transition on the letter $\langle a \rangle$. Since the single state is not final, we have not changed the language of the automaton, which remains empty. However, applying the looping acceptance condition to the original and the modified automata will result in different languages. The following theorem shows that there are automata of the same size as those implementing the current semantics of PSL/SVA, for which a manipulation of this sort works.

Theorem 10 *Let r be a SERE and let φ be a PSC formula.*

- *There exist non-deterministic finite automata (NFW) $\mathcal{N}_{\mathcal{L}}(r)$, $\mathcal{N}_{\mathcal{F}}(r)$ and a Büchi automaton (NBW) $\mathcal{B}_I(r)$ each with $O(2^{|r|})$ states that accept $\mathcal{L}(r)$, $\mathcal{F}(r)$ and $\mathcal{I}(r)$, respectively. Moreover, these automata agree on all components but the set of accepting states.*
- *There exists an alternating Büchi automaton (ABW) B_{φ} with $O(2^{|\varphi|})$ states that accepts exactly the set of words w such that $w \models \varphi$.*
- *The satisfiability and model checking problems for formulas in PSC are EXPSPACE-complete.*

Theorem 10 shows that the proposed semantics is not harder than the current semantics, and thus that complexity is not a motivation for preferring one over the other, or for considering structural contradictions to be satisfiable. This is in contrast to logical contradictions, for which, as shown by [4], complexity is an important consideration.

5 Conclusion and future work

We have presented a semantics that treats logical and structural contradictions in a consistent manner. Finding a solution was complicated by the need to support not only intersection, but also fusion and the three views (which are needed by `abort`). We defined three separate languages for SERES. The language $\mathcal{L}(r)$ is the traditional language of a semi-extended regular expression. The language of proper prefixes of a SERE, denoted $\mathcal{F}(r)$, consists of *finite* proper prefixes of words in $\mathcal{L}(r)$, except that logical and structural contradictions are considered satisfiable. The loop language of a SERE, denoted $\mathcal{I}(r)$, consists of *infinite* words in which we get “stuck forever” in a starred sub-expression of r . A bonus is that our semantics are defined on the natural alphabet 2^P

rather than the inflated alphabet $2^P \cup \{\top, \perp\}$, currently used by both PSL and SVA. The complexity of model checking PSC is not affected by the changes we propose.

Previously [9], we used a topological characterization to prove that without the intersection and fusion operators, the relation between $r!$ and r is the same as that between strong and weak until. Because that characterization is based on the \top, \perp approach, it breaks down when we admit the intersection and fusion operators. Future work is to find a topological characterization for the solution that we have presented here.

References

1. Accellera Property Specification Language Reference Manual Version 1.0, January 2003.
2. Accellera Property Specification Language Reference Manual Version 1.1, June 2004.
3. B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.
4. R. Armoni, D. Bustan, O. Kupferman, and M. Y. Vardi. Resets vs. aborts in linear temporal logic. In H. Garavel and J. Hatcliff, editors, *Proc. TACAS 2003*, LNCS 2619, pages 65–80. Springer, 2003.
5. S. Ben-David, R. Bloem, D. Fisman, A. Griesmayer, I. Pill, and S. Ruah. Automata construction algorithms optimized for PSL (Deliverable 3.2/4). Technical report, Prosyd, 2005.
6. D. Bustan, D. Fisman, and J. Havlicek. Automata construction for PSL. Technical Report MCS05-04, The Weizmann Institute of Science, May 2005.
7. D. Bustan and J. Havlicek. Some complexity results for SystemVerilog assertions. In T. Ball and R. B. Jones, editors, *Proc. CAV 2006*, LNCS 4144, pages 205–218. Springer, 2006.
8. C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer, 2006.
9. C. Eisner, D. Fisman, and J. Havlicek. A topological characterization of weakness. In *Proc. PODC 2005*, pages 1–8. ACM Press, 2005.
10. C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Van Campenhout. Reasoning with temporal logic on truncated paths. In W. A. Hunt, Jr. and F. Somenzi, editors, *Proc. CAV 2003*, LNCS 2725, pages 27–39. Springer, July 2003.
11. C. Eisner, D. Fisman, J. Havlicek, and J. Mårtensson. The \top, \perp approach to truncated semantics. Technical Report 2006.01, Accellera, May 2006.
12. C. Eisner, D. Fisman, J. Havlicek, A. McIsaac, and D. Van Campenhout. The definition of a temporal clock operator. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proc. ICALP 2003*, LNCS 2719, pages 857–870. Springer, 2003.
13. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.
14. D. Fisman. On the characterization of until as a fixed point under clocked semantics. In K. Yorav, editor, *Proc. HVC 2007*, LNCS 4899, pages 19–33. Springer, 2007.
15. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.
16. D. Harel and R. Sherman. Looping vs. repeating in dynamic logic. *Information and Control*, 55(1-3):175–192, 1982.
17. IEEE Standard for Property Specification Language (PSL). IEEE Std 1850-2005, Annex B.
18. IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language. IEEE Std 1800-2005, Annex E.
19. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
20. A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
21. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.