Augmenting a Regular Expression-Based Temporal Logic with Local Variables

Cindy Eisner
IBM Haifa Research Laboratory
Email: eisner@il.ibm.com

Dana Fisman
Hebrew University and
IBM Haifa Research Laboratory
Email: danafi@cs.huji.ac.il

***Abstract*—The semantics of temporal logic is usually defined with respect to a word representing a computation path over a set of atomic propositions. A temporal logic formula does not control the behavior of the atomic propositions, it merely observes their behavior. Local variables are a twist on this approach, in which the user can declare variables local to the formula and control their behavior from within the formula itself.**

Local variables were introduced in 2002, and a formal semantics was given to them in the context of SVA, the assertion language of SystemVerilog, in 2004. That semantics suffers from several drawbacks. In particular, it breaks distributivity of the operators corresponding to intersection and union. In this paper we present a formal semantics for local variables that solves that problem and others, and compare it to the previous solution.

I. Introduction

The semantics of temporal logic is usually defined with respect to a word representing a computation path over a set of atomic propositions. A temporal logic formula does not control the behavior of the atomic propositions, it merely observes their behavior. Local variables are a twist on this approach, in which the user can declare variables local to the formula and control their behavior from within the formula itself.

Local variables were introduced at the same time independently by Havlicek et al. [11] and by Oliveira and Hu [16]. The latter's work was based on work by Seawright and Brewer [18] that introduced the related concept of *action blocks*. Since then, local variables have made their way into SVA, the assertion language of SystemVerilog [2,14]. The semantics of [2,14] consider a temporal logic based on semi-extended regular expressions, which is at the core of the IEEE standard temporal logics SVA [2,14] and PSL [3,13]. In this paper, we term that core logic DRE (where the D stands for *dynamic modalities* and the RE for *regular expressions*) and use it as our base logic in a formal semantics for local variables that solves serious problems with those of [2,14].

The logic DRE is composed of standalone semi-extended regular expressions (SEREs) and the application of dynamic modalities and Boolean operators to them. The set of SEREs is built from atoms which are Boolean expressions and includes the standard operators concatenation, union, intersection and Kleene closure. If r is a SERE then $r!$ is a DRE formula that holds on words containing a non-empty prefix in the language of r. For example:

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

holds on any word containing the letter a followed by some number of b's followed by some number of c's and then a d.

There are two dynamic modalities in DRE: $\mapsto$ (*suffix implication*) and its dual $\diamondsuit\!\!\rightarrow$ (*suffix conjunction*). If r is a SERE and φ is a DRE formula then $r \mapsto \varphi$ and $r \diamondsuit\!\!\rightarrow \varphi$ are DRE formulas.[1] The formula $r \mapsto \varphi$ holds on a word w if for every prefix v of w that is in the language of r, the suffix of w starting from the last letter of v satisfies φ; the formula $r \diamondsuit\!\!\rightarrow \varphi$ holds on a word w if there exists a prefix v of w that is in the language of r and the suffix of w starting from the last letter of v satisfies φ. For example, consider the following DRE formula:

$$\{\mathit{true}^* \cdot a \cdot\ b^* \cdot\ c^* \cdot\ d\} \mapsto \{e{=}6\}! \tag{2}$$

Informally, Formula 2 holds if whenever we see a word in the language of the SERE $\{a \cdot b^* \cdot c^* \cdot d\}$, we get that $e{=}6$ holds on the final letter of that word.

If we want to see $e{=}6$ only when the number of b's and c's is equal, DRE extended with local variables would allow us to do so like this:[2]

$$\begin{aligned}&\{\mathit{new}(i{\leftarrow}0, j{\leftarrow}0)\\ &\quad \mathit{true}^* \cdot a \cdot (b, i{+}{+})^* \cdot\ (c, j{+}{+})^* \cdot (d \wedge i{=}j)\} \mapsto \{e{=}6\}!\end{aligned} \tag{3}$$

Note that we use $\leftarrow$ for local variable assignment, and $=$ for equality. That is, $i{\leftarrow}0$ is an assignment while $e{=}6$ is a Boolean expression that uses the equality operator. We use $i{+}{+}$ and i -- as abbreviations for the assignments $i{\leftarrow}i+1$ and $i{\leftarrow}i-1$, respectively. In Formula 3, local variables i and j are initialized to 0, and are incremented when we see a b or a c, respectively. At the occurrence of d, i is compared to j, and a word is in the language of the left-hand SERE only if they are equal.

The "locality" of a local variable stems from the fact that different "matches" of the same SERE can be considered to have different "copies" of the local variables. This is similar to programming languages in which each invocation of a function has its own copy of a local variable, even if those invocations

[1] In Dynamic Logic [8,10], $r \mapsto \varphi$ and $r \diamondsuit\!\!\rightarrow \varphi$ correspond to $[r]\varphi$ and $\langle r\rangle\varphi$, respectively.

[2] It might seem like we are increasing the expressive power to that of context free languages, but we limit ourselves to finite domains and therefore the expressive power does not increase. For simplicity, in this paper we omit the declaration of the domain and simply assume that it is finite.

occur concomitantly (e.g., in recursion). Consider Formula 3 on a word such that a holds on the first letter and d on the fourth and sixth. If both b and c hold on all letters, then there are two ways to satisfy the formula by "matching" the left operand of the $\mapsto$. In one "match" we see one b and one c, thus at the occurrence of d we have that $i{=}j{=}1$. In another "match" we see two b's and two c's, thus at the occurrence of d we have that $i{=}j{=}2$.

It thus follows that on a word w such that a and b hold on the first letter and do not hold on any other letter, we expect that the following formulas should not hold:[3]

$$(\mathit{new}(v{\leftarrow}0)\ \{(a, v{+}{+})^* \cdot (b, v{-}{-})^* \cdot \neg b\} \mapsto \{v{=}\quad 1\}!) \quad (4)$$

$$(\mathit{new}(v{\leftarrow}0)\ \{(a, v{+}{+})^* \cdot (b, v{-}{-})^* \cdot \neg b\} \mapsto \{v{=}-1\}!) \quad (5)$$

This, because the left-hand SERE may match the first letter either by matching a or by matching b. When matching a, v is assigned 1 and when matching b it is assigned -1. According to the semantics of the suffix implication operator $\mapsto$, for every prefix in the language of the left-hand side there should exist a suffix matching the right-hand side. Since v may be assigned either 1 or -1 on the left-hand side, neither Formula 4 nor 5 should hold on w. The following formula, however, should hold on w:

$$(\mathit{new}(v{\leftarrow}0)\ \{(a, v{+}{+})^* \cdot (b, v{-}{-})^* \cdot \neg b\} \mapsto \{(v{=}1)\ \vee\ (v{=}-1)\}!) \quad (6)$$

Intuitively, local variables can be seen as a mechanism for both declaring quantified propositions [19] and constraining their behavior. While adding quantified propositions to LTL [17] increases its expressive power from ω-star-free regular expressions to ω-regular expressions [19], it does not increase the expressive power of DRE [2,15], which is ω-regular to begin with [1,15].

As stated earlier, the temporal logic DRE is at the core of the IEEE standard assertion languages SVA and PSL. The former standard includes local variables, while the latter is in the process of adding them. The motivation for local variables in these standards is pragmatic: while they do not add expressive power, they can ease formulation and readability. Formal semantics for local variables were first presented in [14, Annex E]. A closely related semantics was presented and its complexity analyzed in [2], where DRE is denoted SVA^{b+i}.

The semantics of [2,14] suffer from major drawbacks, the main one being that the union and intersection operators do not completely conform to the notions of set union and intersection, and thus break distributivity of $\cup$ over $\cap$. For example, as shown by [12, p. 49], the semantics of [14] give that $\{\mathit{true}\ \cup\ \{\mathit{false}\ \cap\ \{(\mathit{true}, v{\leftarrow}1)\}\}$ is not equivalent to $\{\mathit{true}\ \cup\ \mathit{false}\} \cap \{\mathit{true}\ \cup\ \{(\mathit{true}, v{\leftarrow}1)\}\}$, and the same is easily shown for the similar semantics of [2]. We examine this issue in depth in Sections IV and V.

Breaking standard algebraic properties such as distributivity is quite a serious problem, as most tools use heuristics that assume, for instance, that $A \cup \{B \cap C\}$ can be rewritten into $\{A \cup B\} \cap \{A \cup C\}$, and thus breaking distributivity breaks the tools. Of course, tools can be fixed, but even worse is that most users intuitively assume that the standard algebraic properties hold, and use them without thinking. Thus a logic that breaks them is at great risk of being misused.

In this paper we propose a formal semantics for augmenting DRE with local variables that solves the problems of [2,14]. In our semantics the standard operators on semi-extended regular expressions (concatenation, union, intersection and Kleene closure) are exactly as usual, and thus preserve conventional algebraic properties such as distributivity. In our solution, syntax explicitly determines the scope of a local variable and we assume that variables are not used (referenced or assigned) out of scope. Combining an explicit notion of scope with the standard interpretation of the SERE operators gives us a simple, elegant and intuitive semantics of the same complexity as those of [2,14], but without breaking distributivity or other standard algebraic properties.

As mentioned earlier, the addition of local variables to DRE does not increase expressive power, but may ease formulation of properties in certain cases. In this paper we do not motivate the addition of local variables to temporal logic, and it is not our intention to advocate for their use. Rather, we accept that others have found them useful [2, 11, 14, 16, 18] and in particular note that local variables are already a part of SVA and are in the process of being incorporated into PSL. Thus, given the widespread and growing use of these IEEE standard assertion languages, we feel it is important to get the definition of local variables right, and that is the goal we concentrate on in the sequel.

Our proposed solution is quite simple, which is on the one hand a virtue but on the other hand may give the impression that the problem itself was not a difficult one. However, we note that lack of distributivity has been a known issue in SVA since 2004, but never previously solved.

The remainder of this paper is structured as follows. In the following section we present the logic LVDRE — an extension of DRE with local variables. In Section III we show characteristics of our semantics, in Section IV we compare our semantics to previous work. In Section V we discuss in depth the intersection operator, which is the source of the problems of [2,14] and the motivation for our operator *free*. In Section VI we conclude. The proofs of all propositions are given in the full version of the paper.[4]

II. The logic DRE augmented with local variables

We now present the logic LVDRE, which is the logic DRE extended with local variables. We define LVDRE formulas with respect to a non-empty set of atomic propositions $\mathcal{P}$, a set of local variables $\mathcal{V}$ with domain $\mathcal{D}$, a set of (not necessarily Boolean) expressions $\mathcal{E}$ over $\mathcal{P} \cup \mathcal{V}$, and a set of Boolean expressions $\mathcal{B} \subseteq \mathcal{E}$. For simplicity we do not specify the syntax

[3]The examples of Formulas 4, 5 and 6 are due to Johan Mårtensson.

[4]The full version of the paper is accessible at http://www.cs.huji.ac.il/~danafi/publications/lv_full.pdf.

*Definition 1 (Semi-extended regular expressions (*SEREs*)):*

- If $b \in \mathcal{B}$ is a Boolean expression, X a sequence of local variables and E a sequence of expressions of the same length as X, then the following are SEREs:

  $\bullet\ b \qquad \bullet\ (b, \mathrm{X} \leftarrow \mathrm{E})$

- If r, r_1, and r_2 are SEREs, X a sequence of local variables and E a sequence of expressions of the same length as X, then the following are SEREs:

  $\bullet\ \lambda \qquad \bullet\ r_1 \cdot r_2 \qquad \bullet\ r_1 \cup r_2 \qquad \bullet\ r_1 \cap r_2 \qquad \bullet\ r^* \qquad \bullet\ \{\mathit{new}(\mathrm{X})\ r\} \qquad \bullet\ \{\mathit{new}(\mathrm{X} \leftarrow \mathrm{E})\ r\} \qquad \bullet\ \{\mathit{free}(\mathrm{X})\ r\}$

*Definition 2 (*LVDRE *formulas):* If φ and ψ are LVDRE formulas, r a SERE, X a sequence of local variables and E a sequence of expressions of the same length as X then the following are LVDRE formulas:

$\bullet\ \neg\varphi \qquad \bullet\ \varphi \wedge \psi \qquad \bullet\ r! \qquad \bullet\ r \mapsto \varphi \qquad \bullet\ r\ \diamondsuit\!\!\rightarrow \varphi \qquad \bullet\ (\mathit{new}(\mathrm{X})\ \varphi) \qquad \bullet\ (\mathit{new}(\mathrm{X} \leftarrow \mathrm{E})\ \varphi)$

Fig. 1: The syntax of LVDRE

of expressions, but rather assume that a set of expressions is given. Furthermore we assume that all local variables share

Definition 3 (Tight satisfaction): Let $\mathrm{Z} \subseteq \mathcal{V}$, and let $\gamma_1 \overset{\mathrm{Z}}{\sim} \gamma_2$ (read "γ_1 agrees with γ_2 relative to Z") denote that for every $z \in \mathrm{Z}$ we have that $z(\gamma_1) = z(\gamma_2)$. The notation $\mathtt{v} \models_{\mathrm{z}} r$ means that $\mathtt{v}$ models tightly r with respect to controlled variables Z.

- $\mathtt{v} \models_{\mathrm{z}} b \iff |\mathtt{v}| = 1$ and $b(\mathtt{v}^0|_{\sigma\gamma}) = \mathtt{T}$ and $\mathtt{v}^0|_{\gamma'} \overset{\mathrm{Z}}{\sim} \mathtt{v}^0|_{\gamma}$
- $\mathtt{v} \models_{\mathrm{z}} b, \mathrm{X}{\leftarrow}\mathrm{E} \iff |\mathtt{v}| = 1$ and $b(\mathtt{v}^0|_{\sigma\gamma}) = \mathtt{T}$ and $\mathtt{v}^0|_{\gamma'} \overset{\mathrm{Z}}{\sim} [\mathrm{X}{\leftarrow}\mathrm{E}](\mathtt{v}^0|_{\sigma\gamma})$

- $\mathtt{v} \models_{\mathrm{z}} \lambda \iff \mathtt{v} = \epsilon$
- $\mathtt{v} \models_{\mathrm{z}} r_1 \cdot r_2 \iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$ and $\mathtt{v}_1 \models_{\mathrm{z}} r_1$ and $\mathtt{v}_2 \models_{\mathrm{z}} r_2$
- $\mathtt{v} \models_{\mathrm{z}} r_1 \cup r_2 \iff \mathtt{v} \models_{\mathrm{z}} r_1$ or $\mathtt{v} \models_{\mathrm{z}} r_2$
- $\mathtt{v} \models_{\mathrm{z}} r_1 \cap r_2 \iff \mathtt{v} \models_{\mathrm{z}} r_1$ and $\mathtt{v} \models_{\mathrm{z}} r_2$
- $\mathtt{v} \models_{\mathrm{z}} r^* \iff$ either $\mathtt{v} = \epsilon$ or $\exists \mathtt{v}_1, \mathtt{v}_2$ such that ($\mathtt{v}_1 \neq \epsilon$ and $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$ and $\mathtt{v}_1 \models_{\mathrm{z}} r$ and $\mathtt{v}_2 \models_{\mathrm{z}} r^*$)

- $\mathtt{v} \models_{\mathrm{z}} \{\mathit{new}(\mathrm{X})\ r\} \iff \mathtt{v} \models_{\mathrm{z}\cup\mathrm{X}} r$
- $\mathtt{v} \models_{\mathrm{z}} \{\mathit{new}(\mathrm{X}{\leftarrow}\mathrm{E})\ r\} \iff$ either ($\mathtt{v} = \epsilon$ and $\epsilon \models_{\mathrm{z}} r$) or $\langle \mathtt{v}^0|_{\sigma}, [\mathrm{X}{\leftarrow}\mathrm{E}](\mathtt{v}^0|_{\sigma\gamma}), \mathtt{v}^0|_{\gamma'} \rangle \mathtt{v}^{1\cdot\cdot} \models_{\mathrm{z}\cup\mathrm{X}} r$
- $\mathtt{v} \models_{\mathrm{z}} \{\mathit{free}(\mathrm{X})\ r\} \iff \mathtt{v} \models_{\mathrm{z}\setminus\mathrm{X}} r$

Fig. 2: The semantics of SEREs

particular, we assume that Boolean disjunction, conjunction and negation behave in the usual manner.

Given a local variable x and an expression e we write $[x{\leftarrow}e](\sigma, \gamma)$ to denote the valuation $\hat{\gamma}$ such that $x(\hat{\gamma}) = e(\sigma, \gamma)$ and for every local variable $v \in \mathcal{V} \setminus \{x\}$ we have that $v(\hat{\gamma}) = v(\gamma)$. Given a sequence of local variable $\mathrm{X} = x_1, \ldots, x_n$ and a sequence of expressions $\mathrm{E} = e_1, \ldots, e_n$ of the same length we write $[x_1{\leftarrow}e_1, \ldots, x_n{\leftarrow}e_n](\sigma, \gamma)$ to denote the recursive application $[x_2{\leftarrow}e_2, \ldots, x_n{\leftarrow}e_n](\sigma, [x_1{\leftarrow}e_1](\sigma, \gamma))$. We write $\mathrm{X}{\leftarrow}\mathrm{E}$ to abbreviate $x_1{\leftarrow}e_1, \ldots, x_n{\leftarrow}e_n$.

Semantics of SEREs The semantics of SEREs is defined with respect to a finite good word over Λ and a set of local variables $\mathrm{Z} \subseteq \mathcal{V}$, and is given in Definition 3 in Figure 2. The role of Z, which we call the set of *controlled variables*, is to support scoping. Any variable in Z (this will correspond to the variables in scope) must keep its value if not assigned and take on the assigned value otherwise, whereas any variable not in Z (this will correspond to the variables not in scope) is free to take on any value.

Examine the light gray section of Definition 3. A SERE b holds on an extended word if the word consists of exactly one letter such that b holds on that letter and the γ' components of the letter record no assignments (that is, the post-value of every local variable in scope is the same as its pre-value). The SERE $(b, \mathrm{X}{\leftarrow}\mathrm{E})$ holds on an extended word if the word consists of exactly one letter such that b holds on that letter, and the γ' components of the letter record the assignments $\mathrm{X}{\leftarrow}\mathrm{E}$.

The dark gray section of Definition 3 shows the operators *new* and *free*, which are used to declare a variable (and thus add it to the current scope) and to free a variable (remove it from the current scope), respectively.

Examine now the medium gray section of Definition 3. The semantics of the empty SERE λ and of SERE concatenation, union, intersection and Kleene closure are exactly as usual.

Semantics of formulas The semantics of formulas, shown in Definition 4 in Figure 3, is defined with respect to a finite/infinite word over Σ, a valuation γ of the local variables and a set $\mathrm{Z} \subseteq \mathcal{V}$ of local variables. The role of Z is to support scoping, exactly as in tight satisfaction. The role of γ is to supply a current valuation of the local variables.

Examine the light gray section of Definition 4. When evaluating a formula containing a SERE over a word w we quantify over enhancements to w with respect to γ, where informally, an extended word $\mathtt{w}$ enhances w with respect to γ, denoted $\mathtt{w} \sqsupset \langle w, \gamma \rangle$, if $\mathtt{w}$ preserves w on $w|_\sigma$ and uses γ as the starting pre-value (this is defined formally in the preamble of the definition). If we are evaluating a formula of the form $r!$ or $r \diamondsuit\!\!\rightarrow \varphi$, then the quantification is existential, to match the existential quantification on prefixes of w in the semantics of DRE without local variables. If we are evaluating a formula of the form $r \mapsto \varphi$, then the quantification is universal, to match the universal quantification on prefixes of w in the semantics of DRE without local variables. For $\mapsto$ and $\diamondsuit\!\!\rightarrow$ we use the post-value of the last letter that "matched" r as the starting valuation of φ. For example, in the following formula:

$$(\mathit{new}(i)\ \{\mathit{true}^* \cdot (a, i{\leftarrow}0) \cdot (b, i{+}{+})^* \cdot c\} \mapsto \{(d, i{-}{-})^* \cdot (e \wedge i{=}0)\}!) \tag{7}$$

the post-value of i at the end of the "match" of the left operand records the number of b's that were seen, and is the starting point for the evaluation of the right operand. Thus, we get that Formula 7 holds on words such that if we see an a followed by some number (call it i) of b's followed by a c, then starting from the letter that "matched" c we should see i d's followed by an e.

Definition 4 (Satisfaction): We say that a word $\mathtt{w}$ over Λ *enhances* $\langle w, \gamma\rangle$, denoted $\mathtt{w} \sqsupset \langle w, \gamma\rangle$, iff $\mathtt{w}$ is good, $\mathtt{w}|_\sigma = w$, and $\mathtt{w}^0|_\gamma = \gamma$. The notation $\langle w, \gamma\rangle \models_{\mathrm{Z}} \varphi$ means that the word w satisfies φ with respect to controlled variables $\mathrm{Z} \subseteq \mathcal{V}$ and current valuation of variables γ.

- $\langle w, \gamma\rangle \models_{\mathrm{Z}} r!$ $\iff$ $\exists \mathtt{w} \sqsupset \langle w, \gamma\rangle$, $j < |w|$ such that $\mathtt{w}^{0..j} \not\models_{\mathrm{Z}} r$
- $\langle w, \gamma\rangle \models_{\mathrm{Z}} r \mapsto \varphi$ $\iff$ $\forall \mathtt{w} \sqsupset \langle w, \gamma\rangle$ and $j < |w|$: if $\mathtt{w}^{0..j} \not\models_{\mathrm{Z}} r$ then $\langle w^{j\cdot\cdot}, \mathtt{w}^j|_{\gamma'}\rangle \models_{\mathrm{Z}} \varphi$
- $\langle w, \gamma\rangle \models_{\mathrm{Z}} r \diamondsuit\!\!\rightarrow \varphi$ $\iff$ $\exists \mathtt{w} \sqsupset \langle w, \gamma\rangle$, $j < |w|$ such that $\mathtt{w}^{0..j} \not\models_{\mathrm{Z}} r$ and $\langle w^{j\cdot\cdot}, \mathtt{w}^j|_{\gamma'}\rangle \models_{\mathrm{Z}} \varphi$

- $\langle w, \gamma\rangle \models_{\mathrm{Z}} \neg\varphi$ $\iff$ $\langle w, \gamma\rangle \not\models_{\mathrm{Z}} \varphi$
- $\langle w, \gamma\rangle \models_{\mathrm{Z}} \varphi \wedge \psi$ $\iff$ $\langle w, \gamma\rangle \models_{\mathrm{Z}} \varphi$ and $\langle w, \gamma\rangle \models_{\mathrm{Z}} \psi$

- $\langle w, \gamma\rangle \models_{\mathrm{Z}} (\mathit{new}(\mathrm{X})\ \varphi)$ $\iff$ $\langle w, \gamma\rangle \models_{\mathrm{Z}\cup\mathrm{X}} \varphi$
- $\langle w, \gamma\rangle \models_{\mathrm{Z}} (\mathit{new}(\mathrm{X}{\leftarrow}\mathrm{E})\ \varphi)$ $\iff$ $\langle w, [\mathrm{X}{\leftarrow}\mathrm{E}](w^0, \gamma)\rangle \models_{\mathrm{Z}\cup\mathrm{X}} \varphi$

Fig. 3: The semantics of LVDRE formulas

The dark and medium gray sections of Definition 4 show the *new* operator, similar to that of tight satisfaction, and the operators $\neg$ and $\wedge$, which have the usual semantics.

Semantics with respect to a model Given a model M and a computation path π of M we use $L(\pi)$ to denote the word over Σ that results from mapping each state in M to a letter in Σ in the obvious way. The notation $M \models \varphi$ means that for every computation path π of M, and every $\gamma \in \Gamma$ we have $\langle L(\pi), \gamma\rangle \models_\emptyset \varphi$. By universally quantifying over all contexts of local variables we consider all possible initial values for them. The set Z is initially empty, as a variable is in scope only if it was declared (and not freed).

Extending the definition of local variables to all of PSL and SVA Our formal semantics is easily extendable from LVDRE to all of PSL and SVA, although the details of doing so are beyond the scope of this paper. The extension to PSL includes LTL-style temporal operators [17], the fusion operator [3,13], weak SEREs [5], clock [7,9] and abort [6] operators and is detailed in [4]. The extension to SVA, which includes a subset of the features in [4], is similar.

III. Characteristics

In this section we state several properties of the proposed semantics (all are proven in the full version of the paper).

Proposition 1: Let r be a SERE and φ be an LVDRE formula. Let $\mathcal{P}$ be the set of atomic propositions in r (resp. φ). Let $\mathcal{V}$ be the set of local variables in r (resp. φ) and let $\mathcal{D}$ be their domain. Let $\Sigma = 2^{\mathcal{P}}$ and $\Gamma = \mathcal{D}^{\mathcal{V}}$. Let $\mathrm{Z} \subseteq \mathcal{V}$ and $\gamma \in \Gamma$.

- There exists a non-deterministic finite automaton (NFW) $N_{\mathrm{Z}\gamma}(r)$ with $O(\mathcal{D}^{|r|\cdot 2|\mathcal{V}|})$ states that accepts exactly the set of words $\mathtt{v}$ such that $\mathtt{v} \not\models_{\mathrm{Z}} r$ and $\mathtt{v}^0|_\gamma = \gamma$.
- There exists an alternating Büchi automaton (ABW) B^γ_φ with $O(\mathcal{D}^{|\varphi|\cdot 2|\mathcal{V}|})$ states that accepts exactly the set of words w such that $\langle w, \gamma\rangle \models_{\mathrm{Z}} \varphi$.

When the domain of local variables is $\{\mathrm{T}, \mathrm{F}\}$ we get that the NFW and ABW are of sizes $O(2^{|r|\cdot 2|\mathcal{V}|})$ and $O(2^{|\varphi|\cdot 2|\mathcal{V}|})$, respectively (the source of the exponent in the size of the SERE/formula is the SERE intersection operator). Thus the automata that we build here are slightly bigger than those of [2] for which there exists an NFW (ABW) recognizing a given SERE (formula) of size $O(2^{|r|\cdot|\mathcal{V}|})$ (resp. $O(2^{|\varphi|\cdot|\mathcal{V}|})$. This is since our automata need to take into account the set Z of controlled local variables. At any rate, the complexity of the satisfiability and model checking problems is exactly as in [2], as stated in Proposition 2. The proof follows that of [2, Theorems 1 and 2].

Proposition 2: The satisfiability and model checking problems for properties in LVDRE are EXPSPACE-complete.

Proposition 3 below shows that our semantics preserves standard algebraic properties and in particular distributivity. Proposition 4 shows that idempotence for union and intersection is exactly as usual and that the identity element for union and concatenation is exactly as usual, while the identity element for intersection (usually *true**) changes slightly under our semantics, because we need to take all local variables out of scope explicitly using *free*.

Proposition 3: The following properties hold for the semantics of LVDRE.

- The operators $\cap$ and $\cup$ are commutative.
- The operators $\cap$, $\cup$, $\cdot$ are associative.
- The operator $\cap$ distributes over the operator $\cup$.
- The operator $\cup$ distributes over the operator $\cap$.

Proposition 4: Let r be a SERE and let $\mathcal{V}$ be the set of local variables. Then the following equivalences hold for the semantics of LVDRE.

- $r \equiv \{r\ \cap\ r\}$
- $r \equiv \{\{\mathit{free}(\mathcal{V})\ \mathit{true}\}^*\ \cap\ r\}$
- $r \equiv \{\lambda \cdot r\} \equiv \{r \cdot \lambda\}$
- $r \equiv \{r \cup r\}$
- $r \equiv \{\mathit{false}\ \cup\ r\}$

IV. Comparison with previous approaches

We now compare our semantics to those of [2,14]. Due to space limitations, we show only the semantics of the SERE intersection operator (given in Figure 4). The comparison is done along two axes, substance and style.

$$w, L_0, L_1 \models r_1 \cap r_2 \iff \exists L', L'' \text{ such that } w, L_0, L' \models r_1 \text{ and } w, L_0, L'' \models r_2 \text{ and } L_1(v) = L'(v) \text{ if } v \text{ is assigned in } r_1 \text{ and } L_1(v) = L''(v) \text{ otherwise}$$

$$w, L_0, L_1 \models r_1 \cap r_2 \iff \exists L', L'' \text{ s.t. } w, L_0, L' \models r_1 \text{ and } w, L_0, L'' \models r_2 \text{ and } L_1 = L'|_{D'} \cup L''|_{D''}, \text{ where } D' = \textbf{\textit{flow}}(dom(L_0), r_1) - (\textbf{\textit{block}}(r_1 \cap r_2) \cup \textbf{\textit{sample}}(r_2)) \quad D'' = \textbf{\textit{flow}}(dom(L_0), r_2) - (\textbf{\textit{block}}(r_1 \cap r_2) \cup \textbf{\textit{sample}}(r_1))$$

Fig. 4: The semantics of [2] (shown in light gray) and of [14] (shown in medium gray) for SERE intersection

$$\begin{array}{lcl} \bullet\ \textbf{\textit{sample}}(r_1 \cap r_2) & = & \textbf{\textit{sample}}(r_1) \cup \textbf{\textit{sample}}(r_2) \\ \bullet\ \textbf{\textit{block}}(r_1 \cap r_2) & = & \textbf{\textit{block}}(r_1) \cup \textbf{\textit{block}}(r_2) \cup (\textbf{\textit{sample}}(r_1) \cap \textbf{\textit{sample}}(r_2)) \\ \bullet\ \textbf{\textit{flow}}(X, r_1 \cap r_2) & = & (\textbf{\textit{flow}}(X, r_1) \cup \textbf{\textit{flow}}(X, r_2)) - \textbf{\textit{block}}(r_1 \cap r_2) \end{array}$$

Fig. 5: The functions sample, block and flow of [2,14] for SERE intersection

Regarding substance, we first note that the non-standard semantics of the intersection operator in [2,14] gives startlingly unintuitive interpretations in some cases. For example, consider the following SEREs:

$$\{v{=}4\} \cap \{v{=}13\} \tag{8}$$

$$\{r_1 \cdot v{=}4\} \cap \{r_2 \cdot v{=}13\} \tag{9}$$

Intuitively, the language of SERE 8 should be empty, as should the language of SERE 9 for any SEREs r_1 and r_2. In the semantics of [2,14], the language of SERE 8 is indeed empty, but there exist r_1 and r_2 such that the language of SERE 9 is not empty. For example, letting $r_1 = (\textit{true}, v{\leftarrow}4)$ and $r_2 = \textit{true}$ results in a satisfiable SERE which when concatenated with $v{=}4$ gives the following satisfiable SERE:

$$\{\{(\textit{true}, v{\leftarrow}4) \cdot v{=}4\} \cap \{\textit{true} \cdot v{=}13\}\} \cdot v{=}4 \tag{10}$$

For example, SERE 10 is satisfiable by $\langle w, L_0, L_1 \rangle$ such that w is any word of length 2, in L_0 we have that $v{=}13$ and in L_1 we have that $v{=}4$.

Informally, in the semantics of [2,14] assignment to a local variable on one or both operands of an intersection operator "splits" it into two copies. If it is assigned on only one operand, then the value assigned by that operand is the one that "flows out" of the intersection. If both operands make an assignment, then "syntactic restrictions" are intended to prevent the use of the local variable until it has been reassigned. Under our semantics, all of SEREs 8, 9 and 10 are unsatisfiable.

If we omit the $\cap$ operator, our semantics and those of [2,14] are equivalent.[5] For formulas containing the $\cap$ operator, the semantics are different: in our semantics $\cap$ is pure intersection, while [2,14] uses existential quantification in order to get that the same local variable may be assigned different values in each operand of $\cap$. For this reason the semantics of [2,14] break distributivity, while ours does not. Nevertheless, the two semantics are of the same expressive power (because adding local variables with a finite domain does not change the expressive power, which is ω-regular to begin with).

Regarding style, the semantics of [14] are clearly quite complicated (recall that the dark gray section of Figure 4 gives only the semantics of intersection). In addition it is obvious that a violation of the *sample*, *block* and *flow* rules, whose intention is to provide the above mentioned syntactic restrictions, results in a specific truth value rather than an illegal formula. The semantics of [2] (shown in the light gray section of Figure 4) appear to be somewhat simpler. Note however the asymmetry between r_1 and r_2, which according to [2] is supposed to be resolved by the *sample*, *block* and *flow* rules. While intuitively the intention of this is clear, the details are hazy. The stated intent of the restrictions is to prevent a reference to a local variable at places where it does not have a well-defined value. However, the *sample*, *block* and *flow* rules were designed for semantic use (see the dark gray section of Figure 4) and the exact details of their use as syntactic restrictions are not obvious and are never defined.

In our semantics, on the other hand, there is a single, purely syntactic restriction — that a variable is not used out of scope — that is extremely easy to check. The simplicity of our semantics stems from the fact that treatment of local variables is restricted to the operators that explicitly refer to local variables, and the semantics of the SERE operators that correspond to standard operations on automata is exactly as in standard automata theory.

V. Discussion

In this section we explain why we chose to give explicit control of the scope to users (by introducing the *new* and *free* operators) instead of determining it automatically as done in [2,14].

The SERE intersection operator presents a difficulty. On the one hand it should completely preserve the notion of intersection, but on the other hand, doing so seems to restrict the usefulness of local variables. For example, suppose we

[5] To be precise, without $\cap$ and modulo the fact that in our logic the user has to declare the variables, our semantics is equivalent to those of [2]. The semantics of [14] differ subtly from that of [2] for the $\cup$ operator, for a reason related to the sample, block and flow rules; the details are beyond the scope of this paper.

want a SERE whose language includes only words where the number of a's between s and e equals the number of b's between s and e. Trying:

$$\{\mathit{new}(i{\leftarrow}0) \quad \{s\cdot\{\neg a^*\cdot(a,i{+}{+})\}^*\cdot e\}\cap \quad \{s\cdot\{\neg b^*\cdot(b,i{+}{+})\}^*\cdot e\}\ \} \tag{11}$$

gives us a language that includes only words where the a's and b's occur simultaneously. That is, we get the language of $\{s\cdot\{\neg(a\vee b)^*\cdot(a{\wedge}b)\}^*\cdot e\}$ rather than the language we are looking for. If we try to base our solution on

$$\{\ \ \{\mathit{new}(i_1{\leftarrow}0)\ \{s\cdot\{\neg a^*\cdot(a,i_1{+}{+})\}^*\cdot e\}\}\cap \ \{\mathit{new}(i_2{\leftarrow}0)\ \{s\cdot\{\neg\, b^*\cdot(b,i_2{+}{+})\}^*\cdot e\}\}\ \ \} \tag{12}$$

we are left with a SERE where i_1 and i_2 cannot be compared because their scope is disjoint. We might try this:

$$\begin{array}{l}\{\mathit{new}(i_1{\leftarrow}0,i_2{\leftarrow}0)\\ \quad s\cdot\{\ \{\neg a^*\cdot(a,i_1{+}{+})\}^*\cap\\ \qquad \{\neg\, b^*\cdot(b,i_2{+}{+})\}^*\ \ \}\cdot e\wedge(i_1{=}i_2)\}\end{array} \tag{13}$$

but the result is a SERE that does not hold tightly on any word containing at least one a or one b, because the left-hand side of the intersection increments i_1 while the right-hand side does not and vice-versa for i_2.

One way to get what we want is to keep the restricted scope of i_1 and i_2 as in SERE 12 while adding a third local variable whose scope is the whole SERE. Thus,

$$\begin{array}{l}\{\mathit{new}(i)\\ \quad \{\mathit{new}(i_1{\leftarrow}0)\,\{s\cdot\{\neg a^*\cdot(a,i_1{+}{+})\}^*\cdot(e,i{\leftarrow}i_1)\}\}\ \cap\\ \quad \{\mathit{new}(i_2{\leftarrow}0)\,\{s\cdot\{\neg\, b^*\cdot(b,i_2{+}{+})\}^*\cdot(e,i{\leftarrow}i_2)\}\}\}\end{array} \tag{14}$$

gives us what we want. By assigning both i_1 and i_2 to i, we get that the SERE will "match" only if i_1 and i_2 are equal.

Another way is to use the *free* operator. It provides a way to change the context Z by removing a local variable from the set Z, thus "freeing" it to take on any value. Using the *free* operator, we can express our desired property as follows:

$$\begin{array}{l}\{\mathit{new}(i_1{\leftarrow}0,i_2{\leftarrow}0)\\ \quad s\cdot\{\ \{\mathit{free}(i_2)\ \ \neg a^*\cdot(a,i_1{+}{+})\}^*\cap\\ \qquad \{\mathit{free}(i_1)\ \ \neg\, b^*\cdot(b,i_2{+}{+})\}^*\}\ \cdot e\wedge(i_1{=}i_2)\}\end{array} \tag{15}$$

The local variable i_2 is free to take on any value at all in the left operand of the intersection, and in particular the same value as in the right. And vice-versa, i_1 is free to take on any value at all in the right operand, and in particular the same value as in the left.

It might seem at first glance that control over the scope should be taken care of automatically. For example, define Z to be exactly the set of local variables used in the SERE, and ensure that at least one of the operands of $\cap$ "takes responsibility" for the variable, as shown in the alternative semantics of Figure 6 (which uses existential quantification and thus is somewhat similar in spirit to the semantics of [2,14]).

For $r=\{b\}\cap\{(c,v{\leftarrow}3)\}$ the semantics of Figure 6 works as follows. Let the full set of local variables Z be $\{v\}$. Then

$$\mathtt{v}\models_{\mathrm{Z}} r_1\ \cap\ r_2 \iff \exists \mathrm{Z}_1,\,\mathrm{Z}_2 \text{ s.t. } \mathrm{Z}_1\cup\mathrm{Z}_2=\mathrm{Z} \text{ and } \mathtt{v}\models_{\mathrm{Z}_1} r_1 \text{ and } \mathtt{v}\models_{\mathrm{Z}_2} r_2$$

Fig. 6: An alternative semantics for SERE intersection

we can let Z_1 be $\emptyset$ and Z_2 be $\{v\}$. Thus we get that under the context Z, r holds tightly on a word consisting of a single letter $\mathtt{a}$ where both b and c hold, and $v(\mathtt{a}|_{\gamma'})=3$.

The problem with this approach is that it breaks distributivity. Consider the following SEREs (these are the examples used to illustrate the lack of distributivity of [14] as given in [12, p. 49]):

$$\{\mathit{true}\}\cup\{\mathit{false}\ \cap\ \{(\mathit{true},v{\leftarrow}1)\}\} \tag{16}$$

$$\{\mathit{true}\cup\mathit{false}\}\ \cap\ \{\mathit{true}\cup\{(\mathit{true},v{\leftarrow}1)\}\} \tag{17}$$

Let $\mathtt{v}$ be a word consisting of a single letter $\mathtt{a}$ such that $v(\mathtt{a}|_{\gamma})=0$ and $v(\mathtt{a}|_{\gamma'})=1$. Then under the semantics of Figure 6, for the context $\mathrm{Z}=\{v\}$, SERE 17 holds tightly on $\mathtt{v}$, and SERE 16 does not. To see this, note that SERE 17 can choose Z_1 to be $\emptyset$ and Z_2 to be the set $\{v\}$, so both sides of the $\cap$ hold. Whereas SERE 16 must give the full set Z to both sides of the $\cup$, and thus $\mathtt{v}$ satisfies neither $\{\mathit{true}\}$ (because v is not assigned a value by it, yet we have that $v(\mathtt{a}|_{\gamma})$ and $v(\mathtt{a}|_{\gamma'})$ differ), nor $\{\mathit{false}\ \cap\ \{\mathit{true},v{\leftarrow}1\}\}$ (because of course it does not satisfy *false*). We cannot solve this by letting the $\cup$ operator divide the responsibility between its operands in a similar way that $\cap$ does in Figure 6 above. This is since we are going to see either the left side of the $\cup$ operator or the right side, and in both cases we need one operand to take full responsibility.

Note that all semantics (including those of [2,14]) that try to automatically divide the responsibility for the set Z between r_1 and r_2 will have similar problems. This should be clear, because the semantics of Figure 6 is general in that it says simply that there exist suitable Z_1 and Z_2. If there do not exist Z_1 and Z_2 that work for SEREs 16 and 17, then being more specific cannot help.

Since the scoping cannot be determined automatically without breaking important algebraic properties, our logic includes the *new* and *free* operators that allow explicit control of the scope. Having the scoping determined by the syntax, and the SERE operators adhering to their standard semantics, we get a semantics that preserves distributivity as well as other standard algebraic properties. This is of course very important, since we want the user (and the algorithms and tools) to be able to use intuitive and standard rewrites such as $r_1\cup\{r_2\ \cap\ r_3\}\equiv\{r_1\cup r_2\}\ \cap\ \{r_1\cup r_3\}$ without having to stop and consider carefully the implications that would entail with an alternative definition that breaks distributivity such as that of Figure 6 or of [2,14].

We note that the basic utility of local variables is available without the *free* operator. The purpose of the *free* operator is only to provide added flexibility for easy expression of properties such as that expressed by Formula 15. The se-

mantics of [2,14] obtain that flexibility by using non-standard semantics for the intersection operator and so breaking distributivity, whereas we have chosen to preserve distributivity at the expense of giving explicit control of the scope to the user.

VI. Conclusions

We have presented a relatively simple semantics for local variables that solves serious problems in the widely used industry standard assertion language SVA. In SVA, the non-standard semantics of the intersection operator cause distributivity to be broken, so that users of the logic cannot use the standard intuitions such as that $A \cup \{B \cap C\} \equiv \{A \cup B\} \cap \{A \cup C\}$. In our semantics, distributivity and other standard algebraic properties are preserved, so that the user can freely use his or her existing intuitions when working with the logic.

Our solution is based on the understanding that the scope of local variables cannot be determined automatically by the semantics without breaking important algebraic properties (and as a consequence exhibiting unintuitive interpretations in certain cases). We thus require the user to explicitly control the scope and so equip our logic with operators $\mathit{new}(x)$ and $\mathit{free}(x)$ that add and remove x from scope, respectively. The standard operators of semi-extended regular expressions (concatenation, union, intersection, and Kleene closure) have their usual semantics. This separation between the parts of the semantics that explicitly deal with local variables and the usual operators on SEREs is the key to preserving algebraic properties and not breaking intuition.

Our aim was to solve the drawbacks of [2,14] while maintaining the same complexity. Indeed, though the automata constructed for recognizing our semantics are slightly bigger than those of [2], the satisfiability and model checking problems for formulas in LVDRE have the same complexity under our semantics as under the semantics of [2,14].

We remind the reader that our motivation was not to advocate the use of local variables in temporal logic, but rather to fix a serious problem in SVA, and to prevent a similar problem in PSL. Our solution is easily extendable to cover all of the features in these widely used IEEE assertion languages. As shown in [4], our semantics are easily extendable to all of PSL, including LTL-style temporal operators, the fusion operator, weak SEREs, clocks, and the abort operators; the extension to all of SVA, which includes a subset of the features in [4], is similar.

Acknowledgments

We would like to thank Shoham Ben-David, Doron Bustan, John Havlicek, Erich Marschner, Johan Mårtensson, Avigail Orni, Dmitry Pidan and Sitvanit Ruah for many interesting discussions on the subject of local variables. Thanks to the latter three for insightful comments on previous versions of this paper. Finally, special thanks for Avigail Orni for important comments on an early version of the semantics.

References

[1] D. Bustan, D. Fisman, and J. Havlicek. Automata construction for PSL. Technical Report MCS05-04, The Weizmann Institute of Science, May 2005.

[2] D. Bustan and J. Havlicek. Some complexity results for SystemVerilog assertions. In *Proc. CAV 2006*, LNCS 4144, pages 205–218. Springer, 2006.

[3] C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer, 2006.

[4] C. Eisner and D. Fisman. Proposal for extending Annex B of PSL with local variables, procedural blocks, past expressions and clock alignment operators. Technical Report H-0256, IBM, 2008.

[5] C. Eisner, D. Fisman, and J. Havlicek. A topological characterization of weakness. In *Proc. PODC '05*, pages 1–8, New York, NY, USA, 2005. ACM Press.

[6] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Van Campenhout. Reasoning with temporal logic on truncated paths. In *Proc. CAV '03*, LNCS 2725, pages 27–40. Springer-Verlag, July 2003.

[7] C. Eisner, D. Fisman, J. Havlicek, A. McIsaac, and D. Van Campenhout. The definition of a temporal clock operator. In *Proc. ICALP '03*, LNCS 2719, pages 857–870. Springer-Verlag, June 2003.

[8] M. J. Fischer and R. E. Lander. Propositional dynamic logic of regular programs. In *J. Comput. Syst. Sci.*, pages 18(2), 194–211, 1979.

[9] D. Fisman. On the characterization of until as a fixed point under clocked semantics. In *Proc. Haifa Verification Conference*, volume 4899 of *LNCS*, pages 19–33. Springer, 2007.

[10] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[11] J. Havlicek, N. Levi, H. Miller, and K. Shultz. Extended CBV statement semantics, partial proposal presented to the Accellera Formal Verification Technical Committee, April 2002. At http://www.eda.org/vfv/hm/att-0772/01-ecbv_statement_semantics.ps.gz.

[12] J. Havlicek, K. Shultz, R. Armoni, S. Dudani, and E. Cerny. Notes on the semantics of local variables in Accellera SystemVerilog 3.1 concurrent assertions. Technical Report 2004.01, Accellera, May 2004.

[13] IEEE Standard for Property Specification Language (PSL). IEEE Std 1850™-2005.

[14] IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language. IEEE Std 1800™-2005.

[15] M. Lange. Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In *Proc. CONCUR 2007*.

[16] M. T. Oliveira and A. J. Hu. High-level specification and automatic generation of IP interface monitors. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 129–134, New York, NY, USA, 2002. ACM.

[17] A. Pnueli. A temporal logic of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[18] A. Seawright and F. Brewer. High-level symbolic construction technique for high performance sequential synthesis. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 424–428, New York, NY, USA, 1993. ACM.

[19] A. P. Sistla, M. Y. Vardi, and P. L. Wolper. The Complementation Problem for Büchi Automata, with Applications to Temporal Logic. *Theoretical Computer Science*, 49:217–237, 1987.

APPENDIX

Proposition 1, item 1. *Let r be an* SERE *and φ be a* LVDRE *formula. Let $\mathcal{P}$ be the set of atomic propositions in r (resp. φ). Let $\mathcal{V}$ be the set of local variables in r (resp. φ) and let $\mathcal{D}$ be their domain. Let $\Sigma = 2^{\mathcal{P}}$ and $\Gamma = \mathcal{D}^{\mathcal{V}}$. Let $\mathrm{Z} \subseteq \mathcal{V}$ and $\gamma \in \Gamma$. There exists a non-deterministic finite automaton (*NFW*) $N_{\mathrm{Z}_\gamma}(r)$ with $O(\mathcal{D}^{|r|\cdot 2|\mathcal{V}|})$ states that accepts exactly the set of words $\mathfrak{v}$ such that $\mathfrak{v} \models_{\mathrm{Z}} r$ and $\mathfrak{v}^0|_\gamma = \gamma$.*

Proof. Let r, r_1, r_2 be SEREs, $b \in \mathcal{B}$ a boolean expression $e \in \mathcal{E}$ an expression, all over $\mathcal{V} \cup \mathcal{P}$. Let z be a local variable, and Z a set of local variables. Let $\gamma \in \Gamma$ be a valuation of the local variables. The following is an inductive definition of an NFW $N_{\mathrm{Z}_\gamma}(r)$ that accepts all the words $\mathfrak{v}$ over Λ such that $\mathfrak{v} \models_{\mathrm{Z}} r$ and $\mathfrak{v}^0|_\gamma = \gamma$. The states of $N_{\mathrm{Z}_\gamma}(r)$ are triples of the form $\langle q, \gamma, \mathrm{Z}\rangle$ where q corresponds to the traditional state in a construction of an NFW for a regular expression, γ denotes the current valuation of the local variables and Z records the set of controlled variables.[6]

For the base cases we have:

- $N_{\mathrm{Z}_\gamma}(b) = \langle \Lambda, S_b, I_b, \rho_b, F_b\rangle$ where
  - $S_b = \{\langle q_0, \gamma, \mathrm{Z}\rangle\} \cup (\{q_1\} \times \Gamma \times \mathrm{Z})$,
  - $I_b = \{\langle q_0, \gamma, \mathrm{Z}\rangle\}$ and $F_b = (\{q_1\} \times \Gamma \times \mathrm{Z})$
  - $(\langle q_0, \gamma, \mathrm{Z}\rangle, \mathfrak{a}, \langle q_1, \gamma', \mathrm{Z}\rangle) \in \rho_b$ if $\mathfrak{a}|_\gamma = \gamma$, $\mathfrak{a}|_{\gamma'} = \gamma'$, $b(\mathfrak{a}|_\sigma, \mathfrak{a}|_\gamma) = \mathrm{T}$ and $\mathfrak{a}|_{\gamma'} \overset{\mathrm{Z}}{\sim} \mathfrak{a}|_\gamma$
- $N_{\mathrm{Z}_\gamma}(b, \mathrm{X}\leftarrow\mathrm{E}) = \langle \Lambda, S_b, I_b, \rho_{b,\mathrm{X}\leftarrow\mathrm{E}}, F_b\rangle$ where
  - S_b, I_b and F_b are as above and
  - $(\langle q_0, \gamma, \mathrm{Z}\rangle, \mathfrak{a}, \langle q_1, \gamma', \mathrm{Z}\rangle) \in \rho_{b,\mathrm{X}\leftarrow\mathrm{E}}$ if $\mathfrak{a}|_\gamma = \gamma$, $\mathfrak{a}|_{\gamma'} = \gamma'$, $b(\mathfrak{a}|_\sigma, \mathfrak{a}|_\gamma) = \mathrm{T}$ and $\mathfrak{a}|_{\gamma'} \overset{\mathrm{Z}}{\sim} [\mathrm{X}\leftarrow\mathrm{E}](\mathfrak{a}|_\sigma, \mathfrak{a}|_\gamma)$
- $N_{\mathrm{Z}_\gamma}(\lambda) = \langle \Lambda, \{\langle q_0, \gamma, \mathrm{Z}\rangle\}, \{\langle q_0, \gamma, \mathrm{Z}\rangle\}, \emptyset, \{\langle q_0, \gamma, \mathrm{Z}\rangle\}\rangle$

For the inductive steps we have

- $N_{\mathrm{Z}_\gamma}(\{r\}) = N_{\mathrm{Z}_\gamma}(r)$
- $N_{\mathrm{Z}_\gamma}(\{\mathit{new}(\mathrm{x})\ r\}) = N_{\mathrm{Z}\cup\mathrm{x},\gamma}(r)$
- $N_{\mathrm{Z}_\gamma}(\{\mathit{free}(\mathrm{x})\ r\}) = N_{\mathrm{Z}\setminus\mathrm{x},\gamma}(r)$
- $N_{\mathrm{Z}_\gamma}(r_1 \cdot r_2) = N_{\mathrm{Z}_\gamma}(r_1) \odot N_{\mathrm{Z}_\gamma}(r_2)$
- $N_{\mathrm{Z}_\gamma}(r_1 \cup r_2) = N_{\mathrm{Z}_\gamma}(r_1) \Cup N_{\mathrm{Z}_\gamma}(r_2)$
- $N_{\mathrm{Z}_\gamma}(r_1 \cap r_2) = N_{\mathrm{Z}_\gamma}(r_1) \Cap N_{\mathrm{Z}_\gamma}(r_2)$
- $N_{\mathrm{Z}_\gamma}(r^*) = N_{\mathrm{Z}_\gamma}(r)^{\circledast}$

where the operations $\odot, \circledcirc, \Cup, \Cap$ and $^{\circledast}$ are operators on NFWs defined as follows. Assume $N = \langle \Lambda, S, I, \rho, F\rangle$ and $N_i = \langle \Lambda, S_i, I_i, \rho_i, F_i\rangle$ for $i \in \{1, 2\}$. Assume further that the sets S_1 and S_2 are disjoint. Then:

- $N_1 \odot N_2 = \langle \Lambda, S_1 \cup S_2, I', \rho_1 \cup \rho_2 \cup \rho', F_2\rangle$ where
  - $I' = I_1 \cup I_2$ if $I_1 \cap F_1 \neq \emptyset$ and $I' = I_1$ otherwise
  - $\rho' = \left\{ (s_1, \mathfrak{a}, s_2) \;\middle|\; \begin{array}{l} s_2 = \langle q_2, \gamma_2, \mathrm{Z}_2\rangle \in I_2 \text{ s.t.} \\ \gamma_2 = \mathfrak{a}|_{\gamma'} \text{and } \exists s' \in F_1 \text{ s.t.} \\ (s_1, \mathfrak{a}, s') \in \rho_1 \end{array} \right\}$
- $N_1 \Cup N_2 = \langle \Lambda, S_1 \cup S_2, I_1 \cup I_2, \rho_1 \cup \rho_2, F_1 \cup F_2\rangle$
- $N_1 \Cap N_2 = \langle \Lambda, S_1 \times S_2, I_1 \times I_2, \rho', F_1 \times F_2\rangle$ where $\rho' = \{(\langle s_1, s_2\rangle, \mathfrak{a}, \langle s_1', s_2'\rangle) \mid (s_1, \mathfrak{a}, s_1') \in \rho_1 \text{ and } (s_2, \mathfrak{a}, s_2') \in \rho_2\}$
- $N^{\circledast} = \langle \Lambda, S, I, \rho \cup \rho', F \cup I\rangle$ where $\rho' = \{(s, \mathfrak{a}, s') \mid s \in F \text{ and } \exists s'' \in I \text{ s.t. } (s'', \mathfrak{a}, s') \in \rho\}$

□

[6]The differences between this construction and the one of [2] is that (**a**) our automaton runs on extended words (word over $\Sigma \times \Gamma \times \Gamma$) whereas [2] runs on words over Σ, (**b**) we have the third component that records the set of controlled variables, (**c**) [2] makes use of an external function mapping states of the automaton to possible valuations of local variables to resolve the value of a local variable in case it is assigned by both operands of $\cap$ whereas we need no such a function since the resolution is determined by the intersection itself, and (**d**) this construction provides a clear separation between the part dealing with local variables and the standard operations on automata.

Proposition 1, item 2. *Let r be an* SERE *and φ be a* LVDRE *formula. Let $\mathcal{P}$ be the set of atomic propositions in r (resp. φ). Let $\mathcal{V}$ be the set of local variables in r (resp. φ) and let $\mathcal{D}$ be their domain. Let $\Sigma = 2^{\mathcal{P}}$ and $\Gamma = \mathcal{D}^{\mathcal{V}}$. Let $\mathrm{Z} \subseteq \mathcal{V}$ and $\gamma \in \Gamma$. There exists an alternating Büchi automaton (*ABW*) B^{γ}_{φ} with $O(\mathcal{D}^{|\varphi|\cdot 2|\mathcal{V}|})$ states that accepts exactly the set of words w such that $\langle w, \gamma\rangle \models_{\mathrm{z}} \varphi$.*

Proof. The construction follows that of [1, Proposition 3.10], which provides semantics for PSL interpreted over finite as well as infinite words. Thus it uses automata of the form $\langle \Sigma, S, S_0, \delta, F, A\rangle$ where the sets F and A are used to determine acceptance for finite and infinite words, respectively. That construction uses the fact that $r!$ can be given by means of $\diamondsuit\!\!\rightarrow$ as follows: $r! \equiv r \diamondsuit\!\!\rightarrow \mathit{true}$. Thus, we only need to treat $\mapsto$ and $\diamondsuit\!\!\rightarrow$.

In order to account for local variables, the automaton remembers in each state also the current valuation $\gamma \in \Gamma$ and the set of controlled variables Z.

Following [1] we first replace each occurrence of a subformula of the form $r \diamondsuit\!\!\rightarrow \psi$ in φ by $\langle N_{\mathrm{Z}_\gamma}(r), \psi\rangle$. Let S be the set of tuples $(p, \gamma, \mathrm{Z}, \psi)$ where $p \in \{0, 1\}$ is a parity bit, $\gamma \in \Gamma$ is a valuation of the local variables, Z is a set of controlled variables and ψ satisfies at least one of the following: **i.** ψ is a subformula of φ, **ii.** $\psi \in \{\mathrm{TRUE}, \mathrm{FALSE}, \mathrm{NONEMPTY}\}$, **iii.** ψ is of the form $\langle N_{\mathrm{Z}_\gamma}(r)^q, \vartheta\rangle$, where $\langle N_{\mathrm{Z}_\gamma}(r), \vartheta\rangle$ is a subformula of φ, q is a state of the NFW $N_{\mathrm{Z}_\gamma}(r)$ and for an NFW N and a state q of N, we use N^q to denote the automaton obtained from N by replacing the set of initial states with $\{q\}$. Note that if q is the initial state of N, then $N^q = N$. The initial state is $(0, \gamma_0, \emptyset, \varphi)$.

The set A of accepting states is produced by the following rules:
i. A contains all states of the form: $(1, \gamma, \mathrm{Z}, \mathrm{NONEMPTY})$, $(0, \gamma, \mathrm{Z}, \mathrm{TRUE})$, $(1, \gamma, \mathrm{Z}, \mathrm{FALSE})$, $(1, \gamma, \mathrm{Z}, \langle N^q_{\mathrm{Z}_\gamma}, \psi\rangle)$.
ii. $(p, \gamma, \mathrm{Z}, \neg\psi) \in A$ iff $(1 - p, \gamma, \mathrm{Z}, \psi) \in A$.
iii. $(0, \gamma, \mathrm{Z}, \psi \wedge \vartheta) \in A$ iff both $(0, \gamma, \mathrm{Z}, \psi) \in A$ and $(0, \gamma, \mathrm{Z}, \vartheta) \in A$, and $(1, \gamma, \mathrm{Z}, \psi \wedge \vartheta) \in A$ iff either $(1, \gamma, \mathrm{Z}, \psi) \in A$ or $(1, \gamma, \mathrm{Z}, \vartheta) \in A$.
The set F of final states is equal to A.

The transition relation ρ is defined as follows, where $N^q_{\mathrm{Z}_\gamma} = (\Sigma, S_N, \{q\}, \rho_N, F_N)$.

- $\rho((p, \gamma, \mathrm{Z}, \mathrm{NONEMPTY}), \ell) = (p, \gamma, \mathrm{Z}, \mathrm{TRUE})$

- $\rho((p,\gamma,\mathrm{Z},\textsc{true}),\ell) = (p,\gamma,\mathrm{Z},\textsc{true})$
  $\rho((p,\gamma,\mathrm{Z},\textsc{false}),\ell) = (p,\gamma,\mathrm{Z},\textsc{false})$
- $\rho((0,\gamma,\mathrm{Z},\psi\wedge\vartheta),\ell) =$
  $\rho((0,\gamma,\mathrm{Z},\psi),\ell)\wedge\rho((0,\gamma,\mathrm{Z},\vartheta),\ell)$
  $\rho((1,\gamma,\mathrm{Z},\psi\wedge\vartheta),\ell) =$
  $\rho((1,\gamma,\mathrm{Z},\psi),\ell)\vee\rho((1,\gamma,\mathrm{Z},\vartheta),\ell)$
- $\rho((0,\gamma,\mathrm{Z},\neg\psi),\ell) = \rho((1,\gamma,\mathrm{Z},\psi),\ell)$
  $\rho((1,\gamma,\mathrm{Z},\neg\psi),\ell) = \rho((0,\gamma,\mathrm{Z},\psi),\ell)$
- $\rho((0,\gamma,\mathrm{Z},\langle N^q_{\mathrm{Z}\gamma},\psi\rangle),\ell) = \bigvee_{\mathtt{a}\text{ that enhances }\langle\ell,\gamma\rangle}\mathsf{next}_0(q,\mathtt{a})$
  where $\mathsf{next}_0(q,\mathtt{a}) =$

$$\begin{cases}\bigvee_{q'\in\rho_N(q,\mathtt{a})}(0,\langle N^{q'}_{\mathrm{Z}\gamma},\psi\rangle)\vee & \\ \rho((0,\mathtt{a}|_{\gamma'},\mathrm{Z},\psi),\ell) & \text{if } \rho_N(q,\mathtt{a})\cap F_N\neq\emptyset\\ \bigvee_{q'\in\rho_N(q,\mathtt{a})}(0,\langle N^{q'}_{\mathrm{Z}\gamma},\psi\rangle) & \text{otherwise}\end{cases}$$

  $\rho((1,\gamma,\mathrm{Z},\langle N^q_{\mathrm{Z}\gamma},\psi\rangle),\ell) = \bigwedge_{\mathtt{a}\text{ that enhances }\langle\ell,\gamma\rangle}\mathsf{next}_1(q,\mathtt{a})$
  where $\mathsf{next}_1(q,\mathtt{a}) =$

$$\begin{cases}\bigwedge_{q'\in\rho_N(q,\mathtt{a})}(1,\langle N^{q'}_{\mathrm{Z}\gamma},\psi\rangle)\wedge & \\ \rho((1,\mathtt{a}|_{\gamma'},\mathrm{Z},\psi),\ell) & \text{if } \rho_N(q,\mathtt{a})\cap F_N\neq\emptyset\\ \bigwedge_{q'\in\rho_N(q,\mathtt{a})}(1,\langle N^{q'}_{\mathrm{Z}\gamma},\psi\rangle) & \text{otherwise}\end{cases}$$

- $\rho((p,\gamma,\mathrm{Z},\mathit{new}(\mathrm{X})\ \varphi),\ell) = \rho((p,\gamma,\mathrm{Z}\cup\mathrm{X},\varphi),a)$

The correctness of the constructions is proven similar to the correctness of the constructions in the proof of [1, Proposition 3.10]. The main difference with [1, Proposition 3.10] is the conjuncts/disjuncts that quantifies over all letters $\mathtt{a}$ that enhance $\langle\ell,\gamma\rangle$ that appear in the last two bullets. This is since the semantics of $r\mapsto\varphi$ (resp. $r\diamond\!\!\rightarrow\varphi$) universally (resp. existentially) quantifies over all words $\mathtt{w}$ enhancing the given word $\langle w,\gamma\rangle$. In addition the recursive call to ρ in these cases updates the second component to $\mathtt{a}|_{\gamma'}$ i.e. to the evaluation of local variables resulting at the end of the SERE. Note that the NFW itself checks that the enhanced word $\mathtt{w}$ is good. □

Proposition 2. *The satisfiability and model checking problems for formulas in* LVDRE *are EXPSPACE-complete.*

Proof. The proof of the upper bound follows exactly that of [2, Theorem 1] using the first item of Proposition 1 instead of Lemmas 5 and 6 of [2] and the second item of Proposition 1 instead of Lemma 7 of [2]. The proof of the lower bound follows exactly that of [2, Theorem 2]. □

Lemma 1: Let Z be a set of local variables and $z\in\mathrm{Z}$ a local variable. Let r be an SERE such that r does not assign to z nor free z using the *free* operator. Let $\mathtt{v}$ be a good word over Λ such that $\mathtt{v}\models_{\mathrm{Z}} r$. Then $\mathtt{v}^0|_\gamma(z) = \mathtt{v}^{|\mathtt{v}|-1}|_{\gamma'}(z)$.
Proof. By induction on the structure of r.

1) $\mathtt{v}\models_{\mathrm{Z}} b \iff |\mathtt{v}|=1$ and $b(\mathtt{v}^0|_\sigma,\mathtt{v}^0|_\gamma)=\mathrm{T}$ and $\mathtt{v}^0|_{\gamma'}\overset{\mathrm{Z}}{\sim}\mathtt{v}^0|_\gamma$
   Thus for any $z'\in\mathrm{Z}$ we have $\mathtt{v}^0|_\gamma(z) = \mathtt{v}^0|_{\gamma'}(z)$. In particular for the z in question. Since $|\mathtt{v}|-1=0$ we are done.
2) $\mathtt{v}\models_{\mathrm{Z}}(b,\mathrm{X}\leftarrow\mathrm{E}) \iff |\mathtt{v}|=1$ and $b(\mathtt{v}^0|_\sigma,\mathtt{v}^0|_\gamma)=\mathrm{T}$ and $\mathtt{v}^0|_{\gamma'}\overset{\mathrm{Z}}{\sim}[\mathrm{X}\leftarrow\mathrm{E}](\mathtt{v}^0|_\sigma,\mathtt{v}^0|_\gamma)$
   By the definition of $[\mathrm{X}\leftarrow\mathrm{E}](\mathtt{v}^0|_\sigma,\mathtt{v}^0|_\gamma)$ if $\widehat{\gamma} = [\mathrm{X}\leftarrow\mathrm{E}](\mathtt{v}^0|_\sigma,\mathtt{v}^0|_\gamma)$ then for every $v\in\mathcal{V}\setminus\mathrm{X}$ we have $\widehat{\gamma}(v)=\gamma(v)$ and thus we are done.
3) Cases λ, $\{r\}$ are trivial.
4) $\mathtt{v}\models_{\mathrm{Z}}\{\mathit{new}(\mathrm{X})\ r\}\iff\mathtt{v}\models_{\mathrm{Z}\cup\mathrm{X}} r$
   We have increased the set of local variables with respect to which the semantics is tested. In particular z is still a member of this set, thus by the induction we are done.
5) $\mathtt{v}\models_{\mathrm{Z}}\{\mathit{free}(\mathrm{X})\ r\}\iff\mathtt{v}\models_{\mathrm{Z}\setminus\mathrm{X}} r$
   We have decreased the set of local variables with respect to which the semantics is tested. However, z is still a member of this set, thus by induction we are done.
6) Cases $r_1\cdot r_2$, $r_1\circ r_2$, $r_1\cup r_2$, $r_1\cap r_2$ and r^* follow by induction.

□

Lemma 2: Let Z be a set of local variables. Let $\mathtt{w}$ be a finite word over Λ. Then

1) $\mathtt{w}\models_{\mathrm{Z}}\{\mathit{free}(\mathcal{V})\ \mathit{true}\}^*$
2) $\mathtt{w}\not\models_{\mathrm{Z}}\mathit{false}$.

Proof.

1) $\mathtt{w}\models_{\mathrm{Z}}\{\mathit{free}(\mathcal{V})\ \mathit{true}\}^*\iff\mathtt{w}\models_\emptyset \mathit{true}^*$.
   Clearly this holds.
2) Trivial.

□

Proposition 3, item 1. *The operators $\cap$ and $\cup$ are commutative.*
Proof. Clear from the definitions of these operators being symmetric in r_1 and r_2. □

Proposition 3, item 2, part a. *The operators $\cap$ and $\cup$ are associative.*
Proof.

- $\mathtt{v}\models_{\mathrm{Z}} r_1\cup\{r_2\cup r_3\}\iff$
  $\mathtt{v}\models_{\mathrm{Z}} r_1$ or $\mathtt{v}\models_{\mathrm{Z}}\{r_2\cup r_3\}\iff$
  $\mathtt{v}\models_{\mathrm{Z}} r_1$ or $\mathtt{v}\models_{\mathrm{Z}} r_2$ or $\mathtt{v}\models_{\mathrm{Z}} r_3\iff$
  $\mathtt{v}\models_{\mathrm{Z}}\{r_1\cup r_2\}$ or $\mathtt{v}\models_{\mathrm{Z}} r_3\iff$
  $\mathtt{v}\models_{\mathrm{Z}}\{r_1\cup r_2\}\cup r_3$

- $\mathtt{v}\models_{\mathrm{Z}} r_1\cap\{r_2\cap r_3\}\iff$
  $\mathtt{v}\models_{\mathrm{Z}} r_1$ and $\mathtt{v}\models_{\mathrm{Z}}\{r_2\cap r_3\}\iff$
  $\mathtt{v}\models_{\mathrm{Z}} r_1$ and $\mathtt{v}\models_{\mathrm{Z}} r_2$ and $\mathtt{v}\models_{\mathrm{Z}} r_3\iff$
  $\mathtt{v}\models_{\mathrm{Z}}\{r_1\cap r_2\}$ and $\mathtt{v}\models_{\mathrm{Z}} r_3\iff$
  $\mathtt{v}\models_{\mathrm{Z}}\{r_1\cap r_2\}\cap r_3$

□

Proposition 3, item 2, part b. *Concatenation is associative.*

Proof. $\mathtt{v} \models_z r_1 \cdot \{r_2 \cdot r_3\}$

$\iff \exists \mathtt{v}_1, \mathtt{v}_{23}$ s.t. $\mathtt{v} = \mathtt{v}_1\mathtt{v}_{23}$, $\mathtt{v}_1 \models_z r_1$, and $\mathtt{v}_{23} \models_z \{r_2 \cdot r_3\}$

$\iff \exists \mathtt{v}_1, \mathtt{v}_{23}, \mathtt{v}_2, \mathtt{v}_3$ s.t. $\mathtt{v} = \mathtt{v}_1\mathtt{v}_{23}$, $\mathtt{v}_{23} = \mathtt{v}_2\mathtt{v}_3$, $\mathtt{v}_1 \models_z r_1$, and $\mathtt{v}_2 \models_z r_2$ and $\mathtt{v}_3 \models_z r_3$

$\iff \exists \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3$ s.t. $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2\mathtt{v}_3$, $\mathtt{v}_1 \models_z r_1$, and $\mathtt{v}_2 \models_z r_2$ and $\mathtt{v}_3 \models_z r_3$

$\iff \exists \mathtt{v}_{12}, \mathtt{v}_1, \mathtt{v}_2, \mathtt{v}_3$ s.t. $\mathtt{v}_{12} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v} = \mathtt{v}_{12}\mathtt{v}_3$, $\mathtt{v}_1 \models_z r_1$, and $\mathtt{v}_2 \models_z r_2$ and $\mathtt{v}_3 \models_z r_3$

$\iff \exists \mathtt{v}_{12}, \mathtt{v}_3$ s.t. $\mathtt{v} = \mathtt{v}_{12}\mathtt{v}_3$, $\mathtt{v}_{12} \models_z \{r_1 \cdot r_2\}$, and $\mathtt{v}_3 \models_z r_3$

$\iff \mathtt{v} \models_z \{r_1 \cdot r_2\} \cdot r_3$

□

Proposition 3, item 3. *The operator* $\cap$ *distributes over the operator* $\cup$.

Proof. $\mathtt{v} \models_z \{r_1 \cap r_2\} \cup r$

$\iff \mathtt{v} \models_z \{r_1 \cap r_2\}$ or $\mathtt{v} \models_z r$

$\iff (\mathtt{v} \models_z r_1$ and $\mathtt{v} \models_z r_2)$ or $\mathtt{v} \models_z r$

$\iff (\mathtt{v} \models_z r_1$ or $\mathtt{v} \models_z r)$ and $(\mathtt{v} \models_z r_2$ or $\mathtt{v} \models_z r)$

$\iff \mathtt{v} \models_z \{r_1 \cup r\} \cap \{r_2 \cup r\}$

□

Proposition 3, item 4. *The operator* $\cup$ *distributes over the operator* $\cap$.

Proof. $\mathtt{v} \models_z \{r_1 \cup r_2\} \cap r$

$\iff \mathtt{v} \models_z \{r_1 \cup r_2\}$ and $\mathtt{v} \models_z r$

$\iff (\mathtt{v} \models_z r_1$ or $\mathtt{v} \models_z r_2)$ and $\mathtt{v} \models_z r$

$\iff (\mathtt{v} \models_z r_1$ and $\mathtt{v} \models_z r)$ or $(\mathtt{v} \models_z r_2$ and $\mathtt{v} \models_z r)$

$\iff \mathtt{v} \models_z \{r_1 \cap r\} \cup \{r_2 \cap r\}$

□

Proposition 4, items 1 and 2.

- $r \equiv \{r \cap r\}$
- $r \equiv \{r \cup r\}$

Proof. Immediate from the definition. □

Proposition 4, items 3 and 4.

- $r \equiv \{\{\mathit{free}(\mathcal{V})\ \ \mathit{true}\}^* \cap r\}$
- $r \equiv \{\mathit{false} \cup r\}$

Proof. Let $\mathtt{w}$ be a word over Λ. By Lemma 2 we have that $\mathtt{w} \models_z \{\mathit{free}(\mathcal{V})\ \ \mathit{true}\}^*$ and $\mathtt{w} \not\models_z \mathit{false}$.

- $\mathtt{w} \models_z \{\{\mathit{free}(\mathcal{V})\ \ \mathit{true}\}^* \cap r\}$ iff $\mathtt{w} \models_z \{\mathit{free}(\mathcal{V})\ \ \mathit{true}\}^*$ and $\mathtt{w} \models_z r$ iff $\mathtt{w} \models_z r$.
- $\mathtt{w} \models_z \{\mathit{false} \cup r\}$ iff $\mathtt{w} \models_z \mathit{false}$ or $\mathtt{w} \models_z r$ iff $\mathtt{w} \models_z r$.

□

Proposition 4, item 5. $r \equiv \{\lambda \cdot r\} \equiv \{r \cdot \lambda\}$

Proof. $\mathtt{v} \models_z \{\lambda \cdot r\}$

$\iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 \models_z \lambda$, and $\mathtt{v}_2 \models_z r$

$\iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 = \epsilon$, and $\mathtt{v}_2 \models_z r$

$\iff \mathtt{v} \models_z \{r\}$

$\iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 \models_z r$, and $\mathtt{v}_2 = \epsilon$

$\iff \exists \mathtt{v}_1, \mathtt{v}_2$ such that $\mathtt{v} = \mathtt{v}_1\mathtt{v}_2$, $\mathtt{v}_1 \models_z r$, and $\mathtt{v}_2 \models_z \lambda$

$\iff \mathtt{v} \models_z \{r \cdot \lambda\}$

□

2008-08-03 13:36