Temporal Logics for Concurrent Recursive Programs*

Aiswarya Cyriac
Under the guidance of
Benedikt Bollig, Paul Gastin and Marc Zeitoun

Laboratoire Spécification et Vérification
École Normale Supérieure de Cachan, France

Report of the M2(MPRI) research internship, February–July, 2010

Introduction

Background: Temporal logics enjoy a prime position in the verification of programs, due to the ability to express properties of systems in a natural way and the low complexity of decision procedures. Temporal logics over traces have been well studied and used for verification of concurrent programs[DR95]. Temporal logics over nested words (words with a binary nesting relation which matches, say, a push operation with the corresponding pop operation) have been studied for verification of recursive programs [AAB+08].

Problem Statement: Temporal logics for concurrent recursive programs is naturally the next question to ask, and that is what we consider here. Nested traces (traces with multiple nesting relations, as each nesting relation takes care of a single stack or process) have been studied as a model for concurrent recursive programs in [BGH09], and temporal logics for the same was posed as a future work.

Results: We adapt the techniques in [GK03] to get a general procedure for the satisfiability-checking of temporal logics over nested words if the modalities are defined in monadic second order logic. Our construction gives an ExpTime procedure for satisfiability and model checking. The procedure is indeed optimal. Though the ExpTime hardness follows from

*This report is written in English as the author knows little French. The inconvenience caused is regretted.

the hardness result for pushdown systems [BEM97], we give a direct proof for a smaller fragment which uses only the pure future unary modalities.

We show that the two variable fragment of first order logic which uses only the successor relation and the nesting relation is undecidable over nested words with at least two stacks. To obtain decidability, we adopt the restrictions on the nested words followed in [LTMP07, BGH09]. We define a temporal logic over such restricted traces and show that it can be decided in 2-ExpTime. In fact, the technique in [GK03] goes through to yield 2-ExpTime decidability of any temporal logic with MSO definable modalities.

We disprove an erroneous claim in [AAB$^+$08]. We prove the expressive incompleteness of the pure future fragment of an expressive complete temporal logic for nested words at an arbitrary position.

We give a characterization of the regular languages of nested words by means of recognition by morphisms to suitable algebraic structure. We also introduce regular expressions for nested words and prove that star free regular expressions correspond to first order logic over nested words. The same technique can be used to get the equivalence between star free forest expressions and first order logic over ordered unranked trees ([BW07, Boj07]).

Future directions: Though we have nice complexity results, we do not have an expressive complete temporal logic for concurrent recursive programs. We would like to have an algebraic framework for nested traces as well. We hope the algebraic characterization will help us get an expressively complete logic for nested traces.

Organization: The rest of this report is organized as follows. Section 1 introduces nested words and the temporal logics for nested words along with complexity results. Section 2 introduces nested traces and proves the undecidability of FO^2 over two stack nested traces. A temporal logic NWTrL is also introduced. Section 3 discusses the expressive completeness of temporal logic over nested words. Section 4 gives the algebraic characterization of the regular languages over nested words. Section 5 describes the nested word expressions and shows some closure properties of regular languages of nested words. It shows the equivalence of star free regular expressions to first order logic, over nested words and also over ordered unranked trees.

Acknowledgement

I would like to thank my advisers Benedikt Bollig, Paul Gastin and Marc Zeitoun for the constant support and guidance throughout the course of this internship.

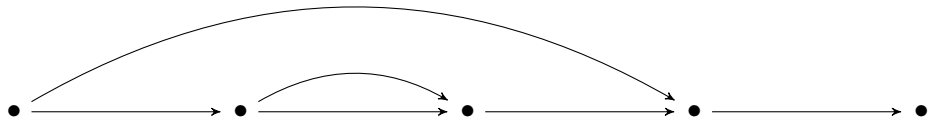

Figure 1: A complete and well-nested word

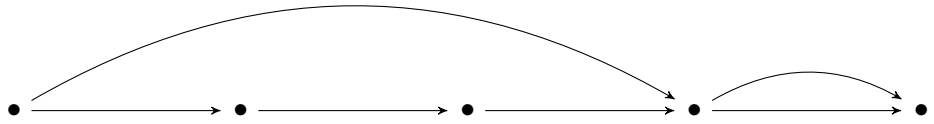

Figure 2: A nested word which is not well-nested

1 Nested words

A **nested word** is a word with additional nesting edges. Nesting edge matches the hierarchically related positions like a call to a subroutine and the return from the subroutine. The recursive behavior is captured using a stack, and hence the nesting edge can be thought of as matching the push operation with the corresponding pop. In this section, systems with only one stack are dealt with. The nesting edges are always from left to right and they do not cross each other. Two nesting edges never share a position. Such nested words are called well-nested. Figure 1 and Figure 3 show examples of well nested words and Figure 2 shows a nested word which is not well-nested.

The positions of the nested word where a nesting edge begins are called *call* positions, and the positions where a nesting edge ends are called *return* positions. We use a call-return alphabet $\Sigma = \mathfrak{C} \cup \mathfrak{R} \cup \mathfrak{I}$ to capture the nesting information. For a nested word $w \in \Sigma^*$, the call positions are precisely the ones labeled by $\mathfrak{C}$, the return positions are the ones labeled by $\mathfrak{R}$ and the positions labeled by $\mathfrak{I}$ are called *internal* positions. Let $\mathfrak{C}(i)$ denote that i is a call position, $\mathfrak{R}(i)$ denote that i is a return position. Let $\mu(i, j)$ denote the existence of a nesting edge from position i to position j. A nested word

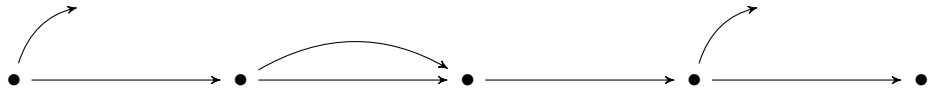

Figure 3: Well nested but not complete

$w \in \Sigma^*$ is well nested if and only if the following conditions hold:

1. $\mu(i,j) \Rightarrow \mathfrak{C}(i) \wedge \mathfrak{R}(j) \wedge i < j$
2. $\mu(i,j) \wedge \mu(i',j') \Rightarrow (i = i' \Leftrightarrow j = j')$
3. $i < j \wedge \mathfrak{C}(i) \wedge \mathfrak{R}(j) \Rightarrow \exists i \leq k \leq j.\, \mu(i,k) \vee \mu(k,j)$

A well nested word is complete if there are no pending calls or returns. Figure 1 shows a complete and well-nested word whereas Figure 3 shows a well-nested word which is not complete. In addition to the above conditions it should satisfy

4. $\mathfrak{C}(i) \Rightarrow \exists j.\mu(i,j)$
5. $\mathfrak{R}(j) \Rightarrow \exists i.\mu(i,j)$

Regularity of nested word languages is captured by recognition by a nested word automaton. A **nested word automaton** is a tuple $(Q, Q_0, Q_f, \delta, \Sigma)$ where Q is the finite set of states, $Q_0 \subseteq Q$ is the set of initial states, and $Q_f \subseteq Q$ is the set of final states. The alphabet is $\Sigma = \mathfrak{C} \cup \mathfrak{R} \cup \mathfrak{I}$. The transition relation is given by $\delta \subseteq (Q \times \mathfrak{C} \times Q \times Q) \cup (Q \times Q \times \mathfrak{R} \times Q) \cup (Q \times \mathfrak{I} \times Q)$. A run of the automaton on a nested word w labels the edges (both linear and nesting). The initial incoming edge can be labeled by some $q_0 \in Q_0$. If $(q_1, c, q_2, q_3) \in \delta$ for some $c \in \mathfrak{C}$, then at a position labeled by c, if the incoming linear edge is labeled q_1, then the outgoing linear edge can be labeled q_2 and the outgoing nesting edge can be labeled q_3. If $(q_1, q_2, r, q_3) \in \delta$ for some $r \in \mathfrak{R}$, then at a position labeled by r, if the incoming linear edge is labeled q_1 and the incoming nesting edge is labeled q_2, then the outgoing linear edge can be labeled q_3. If $(q_1, a, q_2) \in \delta$ for some $a \in \mathfrak{I}$, then at a position labeled by a, if the incoming linear edge is labeled q_1, then the outgoing linear edge can be labeled q_2. The run is accepting if the final outgoing edge is labeled by some $q_f \in Q_f$.

The results given in this paragraph are well established (see e.g. [AM09]). The emptiness checking of a nested word automation can be done in polynomial time (cubic in the number of states). The nested word automaton can be determinized causing an exponential blow-up. The class of regular nested word languages is closed under Boolean operations. The automaton for union and intersection is obtained by product construction, and that for complement is obtained by determinization followed by switching of final and non-final states.

1.1 MSO and FO over nested words

Monadic Second Order logic over nested words with an alphabet Σ is denoted $\mathrm{MSO}_\Sigma(<, \mu)$. We use x, y etc. to denote first-order variables which vary over the positions of the nested word, and X, Y etc. to denote the second-order

variables which vary over the set of positions. The syntax of $\mathrm{MSO}_\Sigma(<, \mu)$ is given by:

$$\varphi, \varphi' ::= P_a(x) \mid X(x) \mid x < y \mid \mu(x, y) \mid \neg\varphi \mid \varphi \vee \varphi' \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$.

A valuation ν assigns positions to first-order variables and sets of positions to second-order variables. Given a nested word w, and a valuation ν, the semantics is defined as follows:

- $w, \nu \models P_a(x)$ iff the position $\nu(x)$ is labeled a in the nested word w.
- $w, \nu \models X(x)$ iff $\nu(x) \in \nu(X)$
- $w, \nu \models x < y$ iff $\nu(x) < \nu(y)$
- $w, \nu \models \mu(x, y)$ iff there is a nesting edge between $\nu(x)$ and $\nu(y)$ in the nested word w.
- $w, \nu \models \neg\varphi$ iff it is not the case that $w, \nu \models \varphi$
- $w, \nu \models \varphi \vee \varphi'$ iff $w, \nu \models \varphi$ and $w, \nu \models \varphi'$
- $w, \nu \models \exists x.\varphi$ iff there is a valuation ν' that agrees with ν on all variables except x such that $w, \nu' \models \varphi$
- $w, \nu \models \exists X.\varphi$ iff there is a valuation ν' that agrees with ν on all variables except X such that $w, \nu' \models \varphi$

The first order logic over nested words $\mathrm{FO}_\Sigma(<, \mu)$ is defined in the same spirit as MSO; the only difference being the absence of second order variables.

1.2 Temporal logic over nested words

We recall the definitions of the nested word temporal logic NWTL and the syntactic extension NWTL^+ defined in [AAB06].

The presence of nesting edges gives rise to multiple paths between two positions of the nested word. However, we want the formulas to be evaluated on an unambiguously specified path. In order to tackle this, we associate different modalities to different types of paths (the ones which use only linear edges, the ones which use nesting edges as much as possible and so on). Let $Call(i)$ denote the innermost call position such that its corresponding return is after i. We define the different types of paths below:

Linear paths: A sequence of positions $i_0 < \cdots < i_k$ is a linear path between i_0 and i_k if and only if $i_j + 1 = i_{j+1}$ for all $0 \leq j < k$.

Call paths: A sequence of positions $i_0 < \cdots < i_k$ is a call path between i_0 and i_k if and only if $i_j = Call(i_{j+1})$ for all $0 \leq j < k$.

Summary paths: A sequence of positions $i_0 < \cdots < i_k$ is a summary path between i_0 and i_k if and only if it traces out the shortest path in the underlying graph of the nested word.

Summary up paths: A restricted summary path which disallows the use of a linear outgoing edge at a call node.

Summary down paths: A restricted summary path which disallows the use of a linear incoming edge at a return node.

Abstract paths: Intersection of summary up and summary down paths. In other words, we are not allowed to take an outgoing linear edge at a call node or an incoming linear edge at a return node.

Note that a summary path is a concatenation of a summary up path and a summary down path.

The logic NWTL considers until modality only along summary paths. NWTL formulas are given by:

$$\varphi, \varphi' ::= \top \mid a \mid \neg\varphi \mid \varphi \vee \varphi' \mid \mathrm{X}\varphi \mid \mathrm{Y}\varphi \mid \mathrm{X}^{\mu}\varphi \mid \mathrm{Y}^{\mu}\varphi \mid \varphi \,\mathrm{U}^{s}\, \varphi' \mid \varphi \,\mathrm{S}^{s}\, \varphi'$$

The syntactically richer NWTL^{+} formulas are given by:

$$\begin{aligned} \varphi, \varphi' \quad ::= \quad & \top \mid a \mid \neg\varphi \mid \varphi \vee \varphi' \mid \mathrm{X}\varphi \mid \mathrm{Y}\varphi \mid \mathrm{X}^{\mu}\varphi \mid \mathrm{Y}^{\mu}\varphi \mid \varphi \,\mathrm{U}\, \varphi' \mid \varphi \,\mathrm{S}\, \varphi' \\ & \mid \varphi \,\mathrm{U}^{s}\, \varphi' \mid \varphi \,\mathrm{S}^{s}\, \varphi' \mid \varphi \,\mathrm{U}^{su}\, \varphi' \mid \varphi \,\mathrm{S}^{su}\, \varphi' \mid \varphi \,\mathrm{U}^{sd}\, \varphi' \mid \varphi \,\mathrm{S}^{sd}\, \varphi' \\ & \mid \varphi \,\mathrm{U}^{a}\, \varphi' \mid \varphi \,\mathrm{S}^{a}\, \varphi' \mid \varphi \,\mathrm{U}^{c}\, \varphi' \mid \varphi \,\mathrm{S}^{c}\, \varphi' \end{aligned}$$

$\mathrm{X}^{\mu}\varphi$ is true at a position x if x is a call position and φ holds at the corresponding return position. *Linear until* (U) and *linear since* (S) are evaluated over linear paths, *summary until* (U^{s}) and *summary since* (S^{s}) are evaluated over summary paths, (U^{su}) and (S^{su}) over summary up paths, (U^{sd}) and (S^{sd}) over summary down paths, (U^{a}) and (S^{a}) over abstract paths and (U^{c}) and (S^{c}) are evaluated over call paths. The semantics is given in Appendix A.

The pure future fragment of NWTL is written NWTL$^{\text{future}}$ in this report. We recall the expressiveness and complexity results on NWTL and NWTL^{+} from [AAB^{+}08].

Theorem 1 ([AAB^{+}08]) *The logics NWTL, NWTL^{+} and $FO_{\Sigma}(<, \mu)$ over nested words are expressively equivalent.*
That is, NWTL = NWTL^{+} = $FO_{\Sigma}(<, \mu)$

Theorem 2 ([AAB^{+}08]) *The satisfiability and model checking problems for NWTL^{+} are* ExpTime *Complete.*

We adapt the technique in [GK03] to get a general framework for satisfiability of temporal logics over nested words with MSO-definable modalities.

Theorem 3 *The satisfiability problem of temporal logics over nested words with $MSO_\Sigma(<, \mu)$-definable modalities*[1] *is decidable in* ExpTime.

Sketch of Proof. Since the modalities are MSO-definable, we have a nested word automaton for it. We can pre-compute these automaton since there are only finitely many modalities. Given a temporal logic formula ϕ, we have polynomially many subformulas of ϕ. For each of these subformulas, we have a nested word automaton verifying that the top level modality of the subformula is computed correctly over an extended alphabet. The satisfiability checking is essentially done by an emptiness checking of the product automaton of the automaton for each subformula. The size of the product automaton is exponential in the size of the formula, hence satisfiability can be checked in ExpTime.

The detailed proof is given in Appendix B. □

Our construction is rather optimal due to the ExpTime hardness of NWTL, which follows from the ExpTime completeness of LTL over pushdown systems [BEM97]. However, we present a direct proof for a smaller fragment which uses only unary modalities and the pure future fragment.

Theorem 4 *Satisfiability checking of $NWTL^{\text{future}}$ with only unary modalities is* ExpTime *complete.*

Proof. The ExpTime complexity follows from Theorem 3. For the hardness we show a reduction from the following ExpTime hard problem:

> Given a linearly bounded alternating Turing machine $\mathcal{M}$ and an input w, does $\mathcal{M}$ accept w?

An alternating Turing machine $\mathcal{M} = (Q = Q_\exists \biguplus Q_\forall, \Sigma, \delta, q_0, F)$, where the finite set of states Q is partitioned into a set of existential states and a set of universal states. A configuration C of $\mathcal{M}$ on w is a $|w|$-length word

$$C = \alpha \cdot \begin{bmatrix} a \\ q \end{bmatrix} \cdot \beta, \text{ where } \alpha, \beta \in (\Sigma \cup \{\llcorner\})^*, \begin{bmatrix} a \\ q \end{bmatrix} \in (\Sigma \cup \{\llcorner\}) \times Q.$$

The symbol $\llcorner$ denotes an empty tape cell. It means that the Turing machine $\mathcal{M}$ is in state q, the tape head is on letter a, and the tape towards the left of the head has α and towards the right of the head has β. If $q \in Q_\forall$ we call C a *universal* configuration. Similarly, if $q \in Q_\exists$, the configuration C is an *existential* configuration. We say C' is a successor configuration of C if there is a transition $\delta \in \Delta$ such that M can go from C to C' on taking δ.

[1] which does not alter the model by relativization

Without loss of generality we assume that every configuration has at most two successor configurations.

Consider the unfolding of the configuration graph rooted at the initial configuration. If $\mathcal{M}$ accepts w, then there is a finite witness subtree with exactly one successor at an internal node if it is an existential configuration, two successors at a universal configuration and all the leaves accepting. We encode a witness subtree as a nested-word. We follow an infix traversal. For a subtree rooted at C, write 'begin' to denote the beginning of the traversal, followed by C, followed by the encoding of the subtree rooted at its children, followed by 'end' to denote the end of traversal. Nesting relation matches 'begin' with the corresponding 'end'.

Note that the length of a configuration is $|w|$. Hence we can write a unary NWTL$^{\text{future}}$ formula to assert that we have a faithful encoding of the above description. If $w = a_1, \ldots, a_{|w|}$,

$$\Psi = \Psi_{init} \wedge \Psi_{comput} \wedge \Psi_{accept}$$

$$\Psi_{init} = \mathsf{begin} \wedge X \begin{bmatrix} a \\ q \end{bmatrix} \wedge \bigwedge_{i=2}^{|w|} X^i a_i \wedge \bigwedge_{i=|w|+1}^{n} X^i{\sqcup} \wedge X^{n+1}\mathsf{begin} \vee \mathsf{end}$$

Ψ_{comput} will be of the form $G(\mathsf{begin} \wedge X^{n+2}\mathsf{begin} \Rightarrow \phi)$. The premise of the implication says that we are going to traverse an internal node. The formula ϕ says the following. Two successor configurations are the same except for a window of three positions with the tape head at the middle. The relation between these three cells in both the configurations can be encoded as dictated by the transition relation of the Turing machine $\mathcal{M}$. The rest of the positions remain the same between the consecutive configurations. This can be taken care of by a conjunction $\bigwedge_{i+1}^{n} X^i a \Leftrightarrow X^{n+1+i} a$ for each $a \in \Sigma$ for an existential configuration and for a universal node and one of its successors. For the successor configuration related by the nesting edge, it will be $\bigwedge_{i+1}^{n} X^i a \Leftrightarrow X^{n+1} X^{\mu} X^i a$.

$$\Psi_{accept} = G(\mathsf{begin} \wedge X^{n+1}\mathsf{end} \Rightarrow \phi_{acc})$$

The premise of the implication says that we are traversing a leaf node and ϕ_{acc} says that it is an accepting one.

Note that $|\Psi|$ is polynomial in $|\mathcal{M}| + |w|$. Clearly if Ψ is satisfiable, then $\mathcal{M}$ has an accepting run on $|w|$.

□

2 When concurrency gets into picture

We assume familiarity with traces (see e.g. [DR95]). We do not need traces from next section onwards.

2.1 Nested 1-stack traces

Let us consider a hypothetical concurrent system which has only one stack (with the danger of one process being able to pop the contents pushed onto the stack by another process). The behavior of such systems could be captured by traces with nesting edges matching a push operation with the (intended) pop operation. We call such a trace a *nested 1-stack trace*. As before, $\mathfrak{C}(i)$ denote that i is a call position, $\mathfrak{R}(i)$ denote that i is a return position. Let $\mu(i,j)$ denote the existence of a nesting edge from position i to position j. A nested 1-stack trace is *existentially valid* if at least one linearization of the trace is a valid nested word. It is *universally valid* if all the linearizations are valid nested words. For a nested 1-stack trace t, let $Lin(t)$ denote the set of linearizations of t which are valid nested words.

There exists a FO formula $\eta(x,y)$ with two free variables such that for all existentially valid nested-traces t, all nested-words $\bar{w} \in Lin(t)$, and positions p, q, we have $t \models p \leq q$ if and only if $\bar{w} \models \eta(p,q)$. Hence we can reduce the satisfiability problem over existentially valid nested 1-stack traces to that over nested-words as follows. For any MSO-definable temporal logic over nested 1-stack traces, interpret the semantics of the modality names over nested-words considering the above translation. For all formulas ξ of the temporal logic, for all existentially valid nested 1-stack traces t, all nested-words $\bar{w} \in Lin(t)$, and all positions p, we have $t, p \models \xi$ if and only if $\bar{w}, p \models \xi$. Hence, as a corollary of Theorem 3 we get the following theorem:

Theorem 5 *The satisfiability problem of any MSO definable temporal logic over existentially valid nested 1-stack traces is decidable in* ExpTime.

Given a nested 1-stack trace t, for all $\bar{w} \in Lin(t)$, $\bar{w}$ is a valid nested-word if and only if for all positions p, q, $(\mathfrak{C}(p) \wedge \mathfrak{C}(q) \Rightarrow p \leq q \vee q \leq p) \wedge (\mathfrak{R}(p) \wedge \mathfrak{R}(q) \Rightarrow p \leq q \vee q \leq p)$. Let us call the above formula ψ. A formula ξ is satisfiable over universally valid nested 1-stack traces if and only if $\xi \wedge \psi$ is satisfiable over existentially valid nested 1-stack traces. Hence we get

Theorem 6 *The satisfiability problem of any MSO definable temporal logic over universally valid nested 1-stack traces is decidable in* ExpTime.

2.2 Nested traces

We now move on to real systems where each process has a separate stack of its own.

Let $\mathfrak{C}_i$ be the push operations on stack i and $\mathfrak{R}_i$ be the pop operations on stack i. In other words, $\mathfrak{C}_i$ is the call alphabet of process i and $\mathfrak{R}_i$ is the return alphabet of process i. Let $\mathfrak{I}$ be the actions that do not touch any stack, moreover we assume that the processes synchronize on $\mathfrak{I}$. That is to say that the alphabet we work on is $\Sigma = \bigcup_i \mathfrak{C}_i \cup \bigcup_i \mathfrak{R}_i \cup \mathfrak{I}$. The nesting relation for stack i matches $\mathfrak{C}_i$-positions to $\mathfrak{R}_i$-positions such that

the projection of a nested trace to $\mathfrak{C}_i \cup \mathfrak{R}_i \cup \mathfrak{I}$ gives a valid nested word. We write Σ_i for $\mathfrak{C}_i \cup \mathfrak{R}_i \cup \mathfrak{I}$.

But this readily gives undecidability. The two variable fragment of first order logic with only the successor relation ($\lessdot$) and nesting relation (μ) is undecidable even for nested words with more than one nesting relation (let us call them *multiple-nested words*).

Theorem 7 *$FO^2(\lessdot)$ over multiple nested words with at least two nesting relations is undecidable.*

Proof. We give a reduction from Post's Correspondence Problem.

> Given two disjoint alphabets A and B, and two functions $f : A \longrightarrow B^*$ and $g : A \longrightarrow B^*$, is there a word $w \in A^+$ such that $f(w) = g(w)$? (where $f(\epsilon) = \epsilon$ and $f(a \cdot w) = f(a) \cdot f(w)$, and similarly for g)

Let $\Sigma = A \cup B$ and $\overline{\Sigma} = \overline{A} \cup \overline{B}$ where $\overline{X} = \{\, \overline{x} \mid x \in X\}$. For a word $w = c_1 \cdots c_k$, $\overline{w} = \overline{c_1} \cdots \overline{c_k}$. We consider multiple nested words over the alphabet $A \cup B$ with two nesting relations. One nesting relation matches a to $\overline{a}$ for $a \in A$ and the other matches b to $\overline{b}$ for $b \in B$. That is, $\mathfrak{C}_1 = A$, $\mathfrak{R}_1 = \overline{A}$, $\mathfrak{C}_2 = B$, $\mathfrak{R}_2 = \overline{B}$ and $\mathfrak{I} = \emptyset$.

Let v^R denote the reverse of the word v. Let $w = a_1 \cdots a_k$ be a solution to the PCP. It can be encoded as a nested-word as follows.

$$a_1 \cdot f(a_1) \cdots a_k \cdot f(a_k) \cdot \overline{g(a_k)}^R \cdot \overline{a_k} \cdots \overline{g(a_1)}^R \cdot \overline{a_1}$$

with μ relating a_i to $\overline{a_i}$ and b to the corresponding $\overline{b}$ for each $b \in B$

Let $\Sigma(x)$ stand for $\bigvee_{a \in \Sigma} P_a(x)$. We now consider the formula Ψ which defines precisely the solutions encoded as above. It asserts the following:

- $\forall x \forall y.\, x \lessdot y \wedge \Sigma(y) \Rightarrow \overline{\Sigma}(x)$
  $\forall x \forall y.\, x \lessdot y \wedge \overline{\Sigma}(x) \Rightarrow \overline{\Sigma}(y)$
  $\exists x \exists y.\, x \lessdot y \wedge \Sigma(x) \wedge \overline{\Sigma}(y)$
  This says that the word is of the form $\Sigma^+.\overline{\Sigma}^+$.

- $\forall x \exists y.\, \mu(x, y) \vee \mu(y, x)$
  $\forall x \forall y.\, \mu(x, y) \Rightarrow c(x) \Leftrightarrow \overline{c}(y)$ for each $c \in \Sigma$

- For each $a \in A$, if $f(a) = b_1 \cdots b_k$ and k is even,
  $\forall x.\, a(x) \Rightarrow (\exists y.\, x \lessdot y \wedge b_1(y) \wedge\ (\exists x.\, y \lessdot x \wedge\ b_2(x) \cdots (\exists x.\, y \lessdot x \wedge\ b_k(x)$
  $\wedge\ (\exists y.\, x \lessdot y \wedge (\overline{\Sigma}(y) \vee A(y)))) \cdots))$
  For each $\overline{a} \in \overline{A}$, if $\overline{g(a)} = \overline{b_1} \cdots \overline{b_k}$ and k is even,
  $\forall x.\, \overline{a}(x) \Rightarrow (\exists y.\, y \lessdot x \wedge \overline{b_1}(y) \wedge\ (\exists x.\, x \lessdot y \wedge\ \overline{b_2}(x) \cdots (\exists x.\, x \lessdot y \wedge\ \overline{b_k}(x)$
  $\wedge\ (\exists y.\, y \lessdot x \wedge (\Sigma(y) \vee \overline{A}(y)))) \cdots))$
  It is similar when k is odd, except that we swap the variables at the end of the above formula. This says that always a is followed by $f(a)$ and $\overline{a}$ is preceded by $\overline{g(a)}^R$.

Clearly the formula Ψ is satisfiable if and only if PCP has a solution.

□

Hence any reasonably expressive logic over words or traces with multiple nesting is undecidable. Hence we proceed to restrict the structure of the multiple-nested word/trace. We accept the notion of k-phase words [LTMP07] and k-phase traces [BGH09].

2.3 k-phase nested words, Mvpa and temporal logics

A k-phase word is a concatenation of at most k many sub-nested words, where each of the sub-nested word may pop at most one stack. A k-phase trace has at least one linearization which is a k-phase word.

In [LTMP07], authors introduce an automaton called k-multistack visibly pushdown automaton (k-MVPA), which captures MSO over k-phase words (called k-MSO). The emptiness checking of k-MVPA can be done in time exponential in the number of states and doubly exponential in k.

Theorem 8 *The satisfiability problem of k-MSO definable temporal logic over k-phase words is decidable in* 2-ExpTime.

The proof is similar to the proof of the Theorem 3, except that we use k-MVPA instead of nested word automaton. Since the emptiness checking of an exponentially sized k-MVPA can be done in doubly exponential time (it becomes triply exponential only in k, but in our case k is a constant), we get the 2-ExpTime complexity.

Theorem 9 *The satisfiability problem of k-MSO definable temporal logic over k-phase traces is decidable in* 2-ExpTime.

The proof is similar to that of Theorem 5 given in Section 2.1. Now we give a temporal logic for nested traces. We call it Nested Trace Temporal Logic NTrTL. It merges the local temporal logic with process based modalities for traces [Thi94] and NWTL$^+$ for nested words. The syntax is given by

$$\begin{aligned}\varphi, \varphi' \quad ::= \quad & \top \mid a \mid p_i \mid \neg\varphi \mid \varphi \vee \varphi' \mid \mathrm{X}_i\varphi \mid \mathrm{Y}_i\varphi \mid \mathrm{X}_i^{\mu}\varphi \mid \mathrm{Y}_i^{\mu}\varphi \mid \varphi\,\mathrm{U}_i\,\varphi' \mid \varphi\,\mathrm{S}_i\,\varphi' \\ & \mid \varphi\,\mathrm{U}_i^{s}\,\varphi' \mid \varphi\,\mathrm{S}_i^{s}\,\varphi' \mid \varphi\,\mathrm{U}_i^{su}\,\varphi' \mid \varphi\,\mathrm{S}_i^{su}\,\varphi' \mid \varphi\,\mathrm{U}_i^{sd}\,\varphi' \mid \varphi\,\mathrm{S}_i^{sd}\,\varphi' \\ & \mid \varphi\,\mathrm{U}_i^{a}\,\varphi' \mid \varphi\,\mathrm{S}_i^{a}\,\varphi' \mid \varphi\,\mathrm{U}_i^{c}\,\varphi' \mid \varphi\,\mathrm{S}_i^{c}\,\varphi'\end{aligned}$$

The intuitive meaning is that p_i holds at a position if and only if the current letter belongs to Σ_i (or the current vertex is in process i). $\mathrm{X}_i\varphi$ holds at a position if and only if the next Σ_i labeled position (next position at process i) satisfies φ. $\mathrm{X}_i^{\mu}\varphi$ holds at a position if and only if the current position is labeled by $\mathfrak{C}_i$ and the corresponding return position satisfies φ. A position satisfies $\varphi\,\mathrm{U}_i^{s}\,\varphi'$ if φ until φ' holds on a summary path labeled by

Σ_i starting from the current vertex. The rest of the modalities are defined in a similar fashion.

As an example, the formula $(X_j\alpha)\ \mathrm{U}_i\ (X_k\beta)$ says that from process i's perspective, the next position in process j satisfies α until the next position in process k satisfies β. Since the modalities of NTrTL are definable in MSO, by Theorem 9 we have a 2-ExpTime satisfiability problem.

3 Pure future fragment of NWTL and expressive completeness

A formula is pure future if its truth value at a position in a word (or any structure with an underlying linear order, like nested words) depends only on the suffix of the word from that position. Similarly a formula is pure past if its truth value at a position depends only on the prefix of the word till that position. A formula is (semantically) separable if it can be written equivalently as a Boolean combination of pure future and pure past formulas. A syntactically pure future formula uses only pure future modalities. A syntactically separable formula can be written equivalently as a Boolean combination of syntactically pure formulas. Clearly syntactic separation implies semantic separation.

We would like to bring to your attention Proposition 4.10 in [AAB$^+$08].It says that there are FO sentences over nested words that cannot be expressed in NWTL$^{\text{future}}$. The (wrong) proof argues that the formula $\alpha \equiv \mathrm{X}^\mu \top \wedge \mathrm{X}^\mu Y a$ cannot be expressed in NWTL$^{\text{future}}$. The error in the proof went unnoticed as the presence of nesting edges was forgotten while considering equivalence between models. In fact we have a NWTL$^{\text{future}}$ formula equivalent to α :

$$(\textsc{call} \wedge a \wedge \mathrm{X}\textsc{ret}) \vee (\textsc{call} \wedge (\mathrm{X}\,(\,\mathrm{X}\textsc{ret} \Rightarrow \textsc{call})))\ \mathrm{U}^s\ (\neg\textsc{call} \wedge \mathrm{X}\,\textsc{ret} \wedge a))$$

where $\textsc{call} \equiv \bigvee_{a\in\mathfrak{C}} a$ and $\textsc{ret} \equiv \bigvee_{a\in\mathfrak{R}} a$. This makes the proof invalid. We had a couple of vain attempts to prove the expressive completeness of NWTL$^{\text{future}}$. Nevertheless we would like to mention them:

- *Gabbay style separation:* [Gab87] The presence of nesting edges poses some difficulty. We could not find separated formula at a position buried under an unbounded number of nesting edges.

- *Future preserving translation from CXPath to NWTL:* We use an ordered unranked tree encoding of nested words. FO over nested words and FO over ordered unranked trees are expressively equivalent. CXPath is FO complete and has the separation property [Mar04]. But a future preserving translation could not be obtained, especially when traversing a path along the descendant relation.

In fact NWTL$^{\text{future}}$ is strictly less expressive than full NWTL at an arbitrary position (say, one buried under an unbounded number of nesting edges).

Theorem 10 *NWTL is not semantically separable at an arbitrary position, even if we consider only complete well nested models.*

Proof. We get the result even without using a call-return alphabet (the one which distinguishes call and return by the letter). Let $\Sigma = \{a, b\}$. We claim that the formula

$$\top \, \mathrm{S}^s \, (b \wedge \mathrm{X}^\mu b)$$

cannot be separated. Suppose it were expressible as a Boolean combination of formulas $\{\varphi_1, \ldots, \varphi_l\}$ where each φ_i is either pure past or pure future formula.

By $M_1(x) \equiv_k M_2(y)$, we mean a k-round Ehrenfeucht-Fraissé game on structure M_1 starting at position x and on M_2 starting at position y cannot distinguish the two structures. For every finite set of formulas $\{\psi_1, \ldots, \psi_m\}$, there exists an integer k such that $M_1(x) \equiv_k M_2(y) \Rightarrow M_1, x \models \psi_i \Leftrightarrow M_2, y \models \psi_i \, , \forall i \in \{1 \ldots m\}$. We write $M_1 \equiv_k M_2$ to mean that a k-round Ehrenfeucht-Fraissé game cannot distinguish the structures M_1 and M_2, with no restriction on the starting position.

Hence there exists an integer k such that for pure future formulas ψ from $\{\varphi_1, \ldots, \varphi_l\}$, and for all words $v, v' \in \Sigma^*$, $a, v \equiv_k a.v' \Rightarrow a.v, 1 \models \psi \Leftrightarrow a.v', 1 \models \psi$ and for all past formulas ψ' from $\{\varphi_1, \ldots, \varphi_l\}$, and for all words $u, u' \in \Sigma^*$, $u.a \equiv_k u'.a \Rightarrow u.a, |u| + 1 \models \psi' \Leftrightarrow u'.a, |u'| + 1 \models \psi'$. Since each of the φ_i is pure past or pure future, for all Boolean combinations ψ'' of $\{\varphi_1, \ldots, \varphi_l\}$, and all words $u, u', v, v' \in \Sigma^*$, $u.a.v \equiv_k u'.a.v' \Rightarrow u.a.v, |u| + 1 \models \psi'' \Leftrightarrow u'.a.v', |u'| + 1 \models \psi''$.

There exist constants k_1, k_2 with $k_1 > k_2$ such that $a^{k_1} b a^{k_2} \equiv_k a^{k_2} b a^{k_1}$. Let $u = v' = a^{k_2} b a^{k_1}$ and $v = a^{k_1} \mathbf{a}^{k_2}$. Consider the nested words $u.a.v$ and $u.a.v'$ with $\{1, \ldots |u|\}$ being call positions and $\{(|u| + 2) \ldots (|u| + |v| + 1 = |u| + |v'| + 1)\}$ being the return positions. The nesting relation is $\{(|u| + 1 - i, |u| + 1 + i) \mid 1 \leq i \leq |u|\}$. Clearly $u.a.v, |u| + 1 \models \top \, \mathrm{S}^s \, (b \wedge \mathrm{X}^\mu b)$ where as $u.a.v', |u|+1 \not\models \top \mathrm{S}^s (b \wedge \mathrm{X}^\mu b)$. Hence $\top \mathrm{S}^s (b \wedge \mathrm{X}^\mu b)$ is not semantically separable at an arbitrary position.

□

However, for top level positions (positions which correspond to empty stack) or positions with fixed finite number of nesting edges crossing it, we do not know whether the theorem holds. We tried to get an algebraic characterization of FO over nested words in order to understand the situation better. Some results we obtained so are given in the next two sections.

4 Algebraic Characterization

Let NW denote the set of all complete nested words (it does not have pending calls or pending returns). Let $\Box$ be a letter not in Σ. Let $\mathrm{NW}_\Box$ denote the set of complete nested words of the form $\Sigma^*.\mathfrak{C}.\Box.\mathfrak{R}.\Sigma^*$. We call $\overline{\mathrm{NW}} = \mathrm{NW} \cup \mathrm{NW}_\Box$ *extended nested words*. We define two types of concatenations — a linear concatenation (denoted $\cdot$) which is the usual one and a hierarchical concatenation (denoted $\boxdot$) in which the $\Box$ in the first word is replaced by the second word. The operations are partially defined from $\overline{\mathrm{NW}} \times \overline{\mathrm{NW}} \mapsto \overline{\mathrm{NW}}$. The precise type of the concatenations are as follows:

$$\cdot : (\mathrm{NW} \times \mathrm{NW} \mapsto \mathrm{NW}) \cup (\mathrm{NW} \times \mathrm{NW}_\Box \mapsto \mathrm{NW}_\Box) \cup (\mathrm{NW}_\Box \times \mathrm{NW} \mapsto \mathrm{NW}_\Box)$$

$$\boxdot : (\mathrm{NW}_\Box \times \mathrm{NW}_\Box \mapsto \mathrm{NW}_\Box) \cup (\mathrm{NW}_\Box \times \mathrm{NW} \mapsto \mathrm{NW})$$

4.1 Algebraic Structure

We define a class of algebraic structures[2] as follows:

$$\mathfrak{M} = (S, S_\Box, \cdot, \boxdot)$$

where $(S, \cdot)$ is a finite monoid, $S_\Box, \boxdot$ is a finite semigroup. Let $\overline{S}$ denote $S \cup S_\Box$. The partial operations $\cdot, \boxdot : \overline{S} \times \overline{S} \mapsto \overline{S}$ are defined exactly as follows:
$\cdot : (S \times S \mapsto S) \cup (S \times S_\Box \mapsto S_\Box) \cup (S_\Box \times S \mapsto S)$.
$\boxdot : (S_\Box \times S_\Box \mapsto S_\Box) \cup (S_\Box \times S \mapsto S)$.

The following identities hold for all $s, s' \in S$ and $t, t' \in S_\Box$:
$t \cdot s = t'$ where $t' \boxdot s' = (t \boxdot s') \cdot s$ for all $s' \in S$
$s \cdot t = t'$ where $t' \boxdot s' = s \cdot (t \boxdot s')$ for all $s' \in S$
$(s \cdot t) \cdot s' = s \cdot (t \cdot s')$

We call these structures *Forest Algebras* though they are defined slightly differently in [BW07].

Morphisms A morphism from extended nested words to the structure $\mathfrak{M}$ is a partial map $h : \overline{\mathrm{NW}} \mapsto \overline{S}$ defined exactly as $h : (\mathrm{NW} \mapsto S) \cup (\mathrm{NW}_\Box \mapsto S_\Box)$ such that the following conditions hold for all $u, u' \in \overline{\mathrm{NW}}$:
$h(\epsilon)$ is $1_{(S,\cdot)}$
$h(u \cdot u') = h(u) \cdot h(u')$
$h(u \boxdot u') = h(u) \boxdot h(u')$

[2]The algebraic structure we present here is a variant of Forest Algebras introduced in [BW07]. We do not need the 'faithfulness' condition of forest algebra. Moreover, we need one semigroup and one monoid, whereas forest algebras need two monoids

Recognition by Morphism A language $L \subseteq \mathrm{NW}$ is recognized by a morphism if there is a structure $\mathfrak{M}$ and a morphism h such that $L = h^{-1}X$ for some $X \subseteq S$.

Theorem 11 *A nested-word language is regular if and only if it is recognized by a Forest Algebra-morphism.*

Proof. if: Let h be a recognizing morphism of the language to a forest algebra $\mathfrak{M} = (S, S_\Box, \cdot, \boxdot)$. Construct a nested word automata with states $S \cup S_\Box \times \mathfrak{R}$. The initial state is $1_{(S,\cdot)}$ and the set of final states is x. At a call node labeled c in state p, it guesses the label of the matching return r and propagates $p \cdot h(c\Box\mathfrak{R}), r$ along the nesting edge and propagates $h(\epsilon)$ along the linear edge. At a return node labeled r in linear state p, if the hierarchical state is (q, r), it propagates $q \boxdot p$ along the linear edge.

only if: If the nested word language L is regular, there is a nested word automaton $(Q, \Sigma, \delta, q_0, F)$ recognizing it. Consider a forest algebra $\mathfrak{M} = (S, S_\Box, \cdot, \boxdot)$ where $S = 2^{Q^2}$ and $S_\Box = 2^{Q^4}$. For $s, s' \in S$ and $t, t' \in S_\Box$

$$\begin{aligned}
s \cdot s' &= \{(q_1, q_2) \mid \exists q_3.(q_1, q_3) \in s \text{ and } (q_3, q_2) \in s'\} \\
s \cdot t &= \{(q_1, q_2, q_3, q_4) \mid \exists q_5.\, (q_1, q_5) \in s \text{ and } (q_5, q_2, q_3, q_4) \in t\} \\
t \cdot s &= \{(q_1, q_2, q_3, q_4) \mid \exists q_5.\, (q_5, q_4) \in s \text{ and } (q_1, q_2, q_3, q_5) \in t\} \\
t \boxdot t' &= \{(q_1, q_2, q_3, q_4) \mid \exists q_5, q_6.\, (q_1, q_5, q_6, q_2) \in t \text{ and } (q_5, q_2, q_3, q_6) \in t'\}
\end{aligned}$$

The morphism h is defined as follows. For $w, w_1, w_2 \in \mathrm{NW}$,

$$\begin{aligned}
h(w) &= \{(q_1, q_2) \mid q_1 \longrightarrow q_2 \text{ on } w\} \\
h(w_1 c\Box r w_2) &= \{(q_1, q_2, q_3, q_4) \mid \exists q_5, q_6, q_7.\, (q_1, q_5) \in h(w_1) \text{ and } (q_6, q_4) \in h(w_2) \\
&\qquad \text{and } (q_5, c, q_2, q_7) \in \delta \text{ and } (q_3, q_7, r, q_6) \in \delta\}
\end{aligned}$$

$L = h^{-1}(X)$ where $X = \{s \mid s \cap (q_0 \times F) \neq \emptyset\}$

□

5 Nested word expressions

The set of regular languages over $\overline{\mathrm{NW}}$ is the smallest set such that

- $\emptyset, \{a\}, \{c\Box r\}$ are regular languages for $a \in \mathfrak{I}$, $c \in \mathfrak{C}$ and $r \in \mathfrak{R}$.
- It is closed under Boolean operations.
- $L \cdot L'$, $L \boxdot L'$ and $L * L'$ are regular languages where $L * L' \equiv L \cdot L' \cup L \boxdot L'$, for regular languages L, L'.

- L^* is regular where $L^* = \bigcup L * \cdots * L$ for all possible bracketings.

Let $L^{\cdot} = (L \cap \mathrm{NW})^* \cdot L \cdot (L \cap \mathrm{NW})^*$ and $L^{\boxdot} = (L \cap \mathrm{NW}_{\Box})^* \boxdot L$.

Proposition 1 *Recognizable nested-word languages are closed under the 'regular operations' defined above, inverse morphisms and quotients by languages of complete nested words.*

Sketch of Proof. We extend the alphabet by adding all possible $c\Box r$, where each $c\Box r$ is read as a single letter. We define the nested word automaton for the extended alphabet. Assert that a box ($\Box$) appears at most once in any string. The automaton for each operation can be constructed similar to the regular words case. $\Box$

5.1 Forest expressions

We recall the definition of forest expressions from [Boj07]. An ordered unranked forest has a horizontal sibling relation and a vertical descendant relation. A context is a forest with exactly one special place holder (denoted $\Box$). Note that a forest need not have a single root. There is horizontal concatenation denoted '+' in which two forests (or a forest and a context) are merged into one side by side. There is a vertical concatenation denoted '.' in which the place holder of one context is replaced by a forest or another context. The star free forest expressions $\mathcal{F}$ and context expression $\mathcal{C}$ are given by the following grammar:

$$\mathcal{F} ::= \emptyset \mid \epsilon \mid a \mid \mathcal{F} + \mathcal{F} \mid \mathcal{F}^* \mid \mathcal{C}\mathcal{F} \mid \mathcal{F} \vee \mathcal{F} \mid \mathcal{F} \wedge \mathcal{F} \mid \neg\mathcal{F}$$

$$\mathcal{C} ::= \Box \mid a\Box \mid \mathcal{C}\mathcal{C} \mid \mathcal{F} + \mathcal{C} \mid \mathcal{C} + \mathcal{F} \mid \mathcal{C} \vee \mathcal{C} \mid \mathcal{C} \wedge \mathcal{C} \mid \neg\mathcal{C}$$

$\emptyset$ is the empty set which is a forest expression. ϵ is the empty forest (which is omitted in the rest of this report). $\vee$ stands for union and $\wedge$ stands for intersection. $\neg$ stands for complementation.

5.2 Star free forest expressions and First order logic

We give an alternative direct and complete proof for the equivalence between SF forest expressions and FO. We need a splitting lemma analogous to Lemma 3.2 in [DG08]:

Lemma 1 *Let A, B be disjoint sub-alphabets of Σ. Abusing notations, let B^*AB^* denote the set of all contexts and forests with exactly one node labeled by A and every other node labeled by B. For any SF language L over Σ,*

$$L \cap B^*AB^* = \bigvee L_1(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3)$$

where $a \in A$. L_1 and L_4 are star free context expressions over B. At most one of L_2, L_5, L_7, L_6, L_3 is a star free context expression over B. Every other

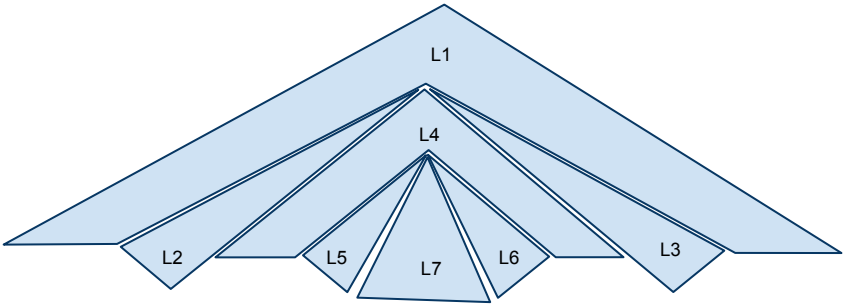

Figure 4: Splitting lemma

L_i is a star free forest expression over B. Moreover we require L_2 and L_3 to be maximal so that the product on the rhs is unambiguous.

Proof. Please refer Figure 4 for a pictorial explanation of the terms. Please refer Appendix C for the detailed proof.

□

Theorem 12 ([Boj07]) *Star free forest expressions have the same expressive power as first order logic.*

Sketch of Proof. The proof is similar to the proof of first order and star free languages over words in [DG08]. We consider an extended alphabet to take care of the free variables. We construct star free expression for the first order formula inductively. The interesting case is the existential quantification. Since we restrict to valid assignments encoded using the extended alphabet, exactly one position will mark the bit for the free variable under consideration. this lets us take an intersection with language of the form B^*AB^*. Due to the splitting lemma we can split this language into a star free expression of star free languages over a smaller sub-alphabet by a bijective renaming. This corresponds to deleting the bit corresponding to the free variable under consideration. □

5.3 Star free Nested word expressions and FO

The splitting lemma for star free nested word expressions is more complicated due the restrictive syntax of the star free nested word expressions. Please see the remark following the lemma for an explanation.

Lemma 2 *Let A, B be disjoint sub-alphabets of Σ. Abusing notations, let B^*AB^* denote the set of all extended nested words with exactly one node*

labeled by A and every other node labeled by B. For any SF language L over Σ,

$$L \cap B^*AB^* = \bigvee_{c\in\mathfrak{C}\cap A, r\in\mathfrak{R}} L_1 \boxdot (L_2 \cdot L_4 \boxdot (L_5 \cdot c\Box r \boxdot L_7 \cdot L_6) \cdot L_3) \quad (1)$$

$$\bigvee_{c\in\mathfrak{C}\cap A, r\in\mathfrak{R}} L_2 \cdot L_4 \boxdot (L_5 \cdot c\Box r \boxdot L_7 \cdot L_6) \cdot L_3 \quad (2)$$

$$\bigvee_{c\in\mathfrak{C}\cap A, r\in\mathfrak{R}} L_5 \cdot c\Box r \boxdot L_7 \cdot L_6 \quad (3)$$

$$\bigvee_{c\in\mathfrak{C}, r\in\mathfrak{R}\cap A} L_1 \boxdot (L_2 \cdot L_4 \boxdot (L_5 \cdot c\Box r \boxdot L_7 \cdot L_6) \cdot L_3) \quad (4)$$

$$\bigvee_{c\in\mathfrak{C}, r\in\mathfrak{R}\cap A} L_2 \cdot L_4 \boxdot (L_5 \cdot c\Box r \boxdot L_7 \cdot L_6) \cdot L_3 \quad (5)$$

$$\bigvee_{c\in\mathfrak{C}, r\in\mathfrak{R}\cap A} L_5 \cdot c\Box r \boxdot L_7 \cdot L_6 \quad (6)$$

$$\bigvee_{a\in\mathfrak{I}\cap A} L_1 \boxdot (L_2 \cdot L_4 \boxdot (L_5 \cdot a \cdot L_6) \cdot L_3) \quad (7)$$

$$\bigvee_{a\in\mathfrak{I}\cap A} L_2 \cdot L_4 \boxdot (L_5 \cdot a \cdot L_6) \quad (8)$$

$$\bigvee_{a\in\mathfrak{I}\cap A} L_5 \cdot a \cdot L_6 \quad (9)$$

$$\bigvee_{c\in\mathfrak{C}\cap A, r\in\mathfrak{R}} L_4 \boxdot (L_5 \cdot c\Box r \cdot L_6) \quad (10)$$

$$\bigvee_{c\in\mathfrak{C}, r\in\mathfrak{R}\cap A} L_5 \cdot c\Box r \cdot L_6 \quad (11)$$

where $a \in A$. $L_1, L_4 \subseteq \mathrm{NW}_\Box$ are star free nested word box expressions over B. At most one of L_2, L_5, L_7, L_6, L_3 is a star free nested word box expression over B. Every other L_i is a star free nested word expression over B. Moreover, we require L_2 and L_3 to be maximal so that the product on the rhs is unambiguous.

Remark. Since a nested word alphabet is typed, for each $a \in A$, the cases when a is a call or return or internal have to be taken care of separately. Since we need the words generated by the regular expression to be complete nested words, we need to consider all possible matching of a given call letter, and hence the product (1). Product (4) is the symmetric case for a return letter. Product (7) deals with the internal letter. Recall that an internal letter do not allow a hierarchical concatenation (or vertical concatenation), and hence the term L_7 is missing in this product. Note that nested word regular expressions do not have $\Box$. Hence products (2), (5) and (8) are special cases of products (1), (4) and (7) respectively with L_1 being the unit

($\Box$). Similarly products (3), (6) and (9) are special cases of products (1),(4) and (7) respectively with L_1 and L_4 being the unit ($\Box$). Products (10) and (11) yield nested word context expressions. The proof of the lemma is given in Appendix D.

Theorem 13 *Star free nested word expressions and first order logic over nested words have the same expressive power.*

Sketch of Proof. Star free to first order is by induction on the structure of the star free expression. Both concatenations require an existential quantification to show the splitting of the concatenation followed by relativization with respect to the newly introduced free variable. First order to star free is similar to the one in [DG08]. $\Box$

Though the proof of both Theorem 12 and Theorem 13 are along the same lines, one does not easily give the other as a corollary.

6 Conclusion and future directions

In this report we show that the temporal logic for nested words and k-phase nested traces have low complexity and that NWTL$^{\text{future}}$ do not have separation property at arbitrary positions. We introduce an algebraic characterization for the regularity of nested word languages. We also introduce nested word expressions and see that first order coincides with star-freeness.

The following are some directions for further research:

- Is there a temporal logic with tolerable complexity for k-phase nested traces with variable k?
- Does NWTL$^{\text{future}}$ have separation property at a top level position?
- Does there exist an expressive complete logic for k-phase nested traces?
- Is there an algebraic framework for nested trace languages?

References

[AAB+08] Rajeev Alur, Marcelo Arenas, Pablo Barceló, Kousha Etessami, Neil Immerman, and Leonid Libkin. First-order and temporal logics for nested words. *LMCS*, 4(4), 2008.

[AM09] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *JACM*, 56(3):1–43, 2009.

[BEM97] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR*, pages 135–150, 1997.

[BGH09] Benedikt Bollig, Manuela-Lidia Grindei, and Peter Habermehl. Realizability of concurrent recursive programs. In *FOSSACS '09*, pages 410–424, Berlin, Heidelberg, 2009. Springer-Verlag.

[Boj07] Mikolaj Bojanczyk. Forest expressions. In *CSL*, pages 146–160, 2007.

[BW07] Mikolaj Bojanczyk and Igor Walukiewicz. Forest algebras. In Jörg Flum, Eric Graedel, and Thomas Wilke, editors, *"Logic and Automata", Texts in Logic and Games*. Amsterdam University Press, 2007.

[DG08] Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1995.

[Gab87] Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification*, pages 409–448, 1987.

[GK03] Paul Gastin and Dietrich Kuske. Satisfiability and model checking for mso-definable temporal logics are in pspace. In *CONCUR*, pages 218–232, 2003.

[LTMP07] Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *LICS*, pages 161–170, Washington, DC, USA, 2007. IEEE Computer Society.

[Mar04] Maarten Marx. Conditional xpath, the first order complete xpath dialect. In *PODS*, pages 13–22, 2004.

[Thi94] P. S. Thiagarajan. A trace based extension of linear time temporal logic. In *LICS*, pages 438–447, 1994.

A Semantics of NWTL$^+$

We recall the syntax of NWTL+ from Section 1.2.

$$\begin{aligned}\varphi, \varphi' \quad ::= \quad & \top \mid a \mid \neg\varphi \mid \varphi \vee \varphi' \mid \mathrm{X}\varphi \mid \mathrm{Y}\varphi \mid \mathrm{X}^{\mu}\varphi \mid \mathrm{Y}^{\mu}\varphi \mid \varphi \,\mathrm{U}\, \varphi' \mid \varphi \,\mathrm{S}\, \varphi' \\ & \mid \varphi \,\mathrm{U}^{s}\, \varphi' \mid \varphi \,\mathrm{S}^{s}\, \varphi' \mid \varphi \,\mathrm{U}^{su}\, \varphi' \mid \varphi \,\mathrm{S}^{su}\, \varphi' \mid \varphi \,\mathrm{U}^{sd}\, \varphi' \mid \varphi \,\mathrm{S}^{sd}\, \varphi' \\ & \mid \varphi \,\mathrm{U}^{a}\, \varphi' \mid \varphi \,\mathrm{S}^{a}\, \varphi' \mid \varphi \,\mathrm{U}^{c}\, \varphi' \mid \varphi \,\mathrm{S}^{c}\, \varphi\end{aligned}$$

The semantics is given below.

- $w, i \models \top$
- $w, i \models a$ iff ith position of the nested word w is labeled a
- $w, i \models \neg\varphi$ iff it is not the case that $w, i \models \varphi$
- $w, i \models \varphi \vee \varphi'$ iff $w, i \models \varphi$ and $w, i \models \varphi'$
- $w, i \models \mathrm{X}\varphi$ iff $w, i+1 \models \varphi$
- $w, i \models \mathrm{Y}\varphi$ iff $w, i-1 \models \varphi$
- $w, i \models \mathrm{X}^{\mu}\varphi$ iff there is a j such that $\mu(i, j)$ holds and $w, j \models \varphi$
- $w, i \models \mathrm{Y}^{\mu}\varphi$ iff there is a j such that $\mu(j, i)$ holds and $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{U}\, \varphi'$ iff there is a linear path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{S}\, \varphi'$ iff there is a linear path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{U}^{s}\, \varphi'$ iff there is a summary path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{S}^{s}\, \varphi'$ iff there is a summary path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{U}^{su}\, \varphi'$ iff there is a summary-up path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{S}^{su}\, \varphi'$ iff there is a summary-up path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{U}^{sd}\, \varphi'$ iff there is a summary-down path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \,\mathrm{S}^{sd}\, \varphi'$ iff there is a summary-down path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$

- $w, i \models \varphi \, \mathrm{U}^a \, \varphi'$ iff there is an abstract path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \, \mathrm{S}^a \, \varphi'$ iff there is an abstract path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$
- $w, i \models \varphi \, \mathrm{U}^c \, \varphi'$ iff there is a call path $i = i_0, i_1, \ldots, i_n$ such that $w, i_n \models \varphi'$ and for all j such that $i_0 \leq j < i_n$, $w, j \models \varphi$
- $w, i \models \varphi \mathrm{S}^c \varphi'$ iff there is a call path $i_0, i_1, \ldots, i_n = i$ such that $w, i_0 \models \varphi'$ and for all j such that $i_0 < j \leq i_n$, $w, j \models \varphi$

Let $F^*\phi$ stand for $\top U^* \phi$ and $P^*\phi$ stand for $\top S^* \phi$. $\top$ is a shorthand for $a \vee \neg a$. *call* and *ret* are abbreviations for $\bigvee_{a \in \mathfrak{C}} a$ and $\bigvee_{a \in \mathfrak{R}} a$ respectively.

B Complexity of temporal logics over nested words

Let B be a set of modality names together with a mapping $arity : B \longrightarrow \mathbb{N}$ giving the arity of each modality. The syntax of the temporal logic $TL(B)$ is given by the grammar

$$\phi ::= \sum_{M \in B} M(\underbrace{\phi, \ldots, \phi}_{arity(M)})$$

Consider the mapping $[\![-]\!] : B \longrightarrow \mathrm{MSO}_\Sigma(<, \mu, call, ret)$ such that if $arity(M) = l$ then $[\![M]\!]$ is an l-ary MSO modality, that is, an MSO formula with l free set variables $X_1, \ldots, X_l$ and one free individual variable x. Given a nested word $\bar{w} = (\, w, \mu, call, ret \,)$ and a formula $\varphi \in TL(B)$, the semantics is given by the set $\varphi^{\bar{w}}$ of positions in w where φ holds. Inductively, if $\varphi = M(\varphi_1, \ldots, \varphi_l)$ where $M \in B$ is of arity $l \geq 0$, then

$$\varphi^{\bar{w}} = \{\, p \leq |w| \mid \bar{w} \models_{MSO} [\![M]\!](\varphi_1^{\bar{w}}, \ldots, \varphi_l^{\bar{w}}, p)\}$$

We also write $\bar{w}, p \models \varphi$ for $p \in \varphi^{\bar{w}}$

We consider the alphabet $\Sigma_l = \Sigma \times \{0,1\}^l$ for $l \in \mathbb{N}$. A letter $a \in \Sigma_l$ will be written $a = (a_0, a_1, \ldots, a_l)$ and a word will be written $w = (w_0, w_1, \ldots, w_l) \in \Sigma^\infty \times (\{0,1\}^\infty)^l$ with $|w| = |w_i|$ for $0 \leq i \leq l$. The support of a word $u \in \{0,1\}^*$ is denoted $supp(u)$ and is defined to be the set of all positions labelled by 1 in the word u. We get the following theorem from the equivalence of MSO over nested-words and nested-word automata [AM09].

Theorem 14 *Let M be an l-ary modality name and let $[\![M]\!]$ be its associated $MSO_\Sigma(<, \mu, call, ret)$-modality. Then there exists a nested word Büchi-automaton B_M over the alphabet Σ_{l+1} such that $\bar{w} \in \mathcal{L}(B_M)$ where $w = (w_0, w_1, \ldots, w_{l+1})$ if and only if $supp(w_{l+1}) = \{\, p \in [|w|] \mid \bar{w}_0 \models_{MSO} [\![M]\!](supp(w_1), \ldots, supp(w_l), p)\}$.*

For formulas φ and ψ, we write $\varphi \leq \psi$ if φ is a subformula of ψ. For $\xi \in TL(B)$, let $Sub(\xi) = \{\, \phi \in TL(B) \mid \phi \leq \xi\}$. We consider the alphabet $\overline{\Sigma} = \Sigma \times \{0,1\}^{Sub(\xi)}$. A word $w \in \overline{\Sigma}^\infty$ is $(w_0, (w_\varphi)_{\varphi \leq \xi})$ where $w_0 \in \Sigma^\infty$, $w_\varphi \in \{0,1\}^\infty$ for $\varphi \leq \xi$ and $|w| = |w_0| = |w_\varphi|$. Let $\psi = M(\varphi_1, \ldots, \varphi_l) \leq \xi$. Then $a \restriction \psi := (a_0, a_{\varphi_1}, \ldots, a_{\varphi_l}, a_\psi) \in \Sigma_{l+1}$. For $w \in \overline{\Sigma}^\infty$, $w \restriction \psi := (w_0, w_{\varphi_1}, \ldots, w_{\varphi_l}, w_\psi) \in \Sigma_{l+1}^\infty$

The construction we present here is similar to the one in [GK03]. For a formula $\varphi \in TL(B)$, let $top(\varphi)$ be the outer most modality name of φ. Let $Q_{top(\varphi)}$ be the set of linear states and $P_{top(\varphi)}$ be the set of hierarchical states of the nested-word Büchi-automaton $\mathcal{B}_{top(\varphi)}$. We construct the automaton $\mathcal{A}_\xi$ such that:

- The set of linear states $Q = \prod_{\varphi \leq \xi} Q_{top(\varphi)}$
- The set of hierarchical states $P = \prod_{\varphi \leq \xi} P_{top(\varphi)}$
- The alphabet is $\overline{\Sigma}$
- For $p = (p_\varphi)_{\varphi \leq \xi}$ and $q = (q_\varphi)_{\varphi \leq \xi}$, $(p, a, q) \in \delta_i$ if and only if for all $\varphi \leq \xi$, we have $(p_\varphi, a \restriction \varphi, q_\varphi) \in \delta_i$ of $\mathcal{B}_{top(\varphi)}$
- For $p = (p_\varphi)_{\varphi \leq \xi}$, $q = (q_\varphi)_{\varphi \leq \xi}$ and $q' = (q'_\varphi)_{\varphi \leq \xi}$, $(p, a, q, q') \in \delta_c$ if and only if for all $\varphi \leq \xi$, we have $(p_\varphi, a \restriction \varphi, q_\varphi, q'_\varphi) \in \delta_c$ of $\mathcal{B}_{top(\varphi)}$
- For $p = (p_\varphi)_{\varphi \leq \xi}$, $p' = (p'_\varphi)_{\varphi \leq \xi}$ and $q = (q_\varphi)_{\varphi \leq \xi}$, $(p, p', a, q) \in \delta_r$ if and only if for all $\varphi \leq \xi$, we have $(p_\varphi, p'_\varphi, a \restriction \varphi, q_\varphi) \in \delta_r$ of $\mathcal{B}_{top(\varphi)}$
- Accepting states are the product of the accepting states of each automation for the finite word case. For the infinite word case we have a Muller condition.

A sequence of linear states $p^0, p^1, \ldots$ and a sequence of hierarchical states $q^0, q^1, \ldots$ define a run of $\mathcal{A}_\xi$ for a nested-word $\bar{w}$ if and only if for each $\varphi \leq \xi$, its projection $p^0_\varphi, p^1_\varphi, \ldots$ and $q^0_\varphi, q^1_\varphi, \ldots$ on φ is run of $\mathcal{B}_{top(\varphi)}$ for the nested-word $\bar{w} \restriction \varphi$. A run of $\mathcal{A}_\xi$ is accepting if and only if for each $\varphi \leq \xi$, its projection on $\mathcal{B}_{top(\varphi)}$ is accepting.

Lemma 3 *Let $w = (w_0, (w_\varphi)_{\varphi \leq \xi})$. Then $\bar{w} = (\, w, \mu, call, ret\,) \in \mathcal{L}(\mathcal{A}_\xi)$ if and only if for each $\varphi \leq \xi$ we have $supp(w_\varphi) = \varphi^{\bar{w}} = \{\, p \in [|w|] \mid \bar{w}, p \models \varphi\}$.*

Lemma 4 *The formula ξ is satisfiable if and only if there exists $\bar{w} \in \mathcal{L}(\mathcal{A}_\xi)$ with $supp(w_\xi) \neq \emptyset$.*

Theorem 15 *Let B be a finite set of modality names with associated MSO-modalities. Then the satisfiability problem for $TL(B)$ is in* ExpTime.

Proof. For each modality $M \in B$, the automata $\mathcal{B}_M$ are fixed and need not be computed. Let $k = \max\{|\mathcal{B}_M| \mid M \in B\}$. Given a $TL(B)$ formula ξ, since $|sub(\xi)|$ is linear in $|\xi|$, we have $|\mathcal{A}_\xi|$ is exponential in $|\xi|$. We restrict the accepted nested-words to the ones with $supp(w_\xi) \neq \emptyset$. For this, we construct $\mathcal{A}'_\xi$ with an additional flag bit in its states. $|\mathcal{A}'_\xi|$ is exponential in $|\xi|$. The emptiness checking of $\mathcal{A}'_\xi$ can be done in polynomial time in the size of $\mathcal{A}'_\xi$ and hence exponential time in $|\xi|$.

□

C Splitting lemma for ordered unranked trees

Proof. The proof is by induction on the structure of the star free expression for L.

- If $L = \{a\}$, $a \notin A$, or if $L = \emptyset$, then $L_7 = \emptyset$.
- If $L = \{a\}$, $a \in A$, then $L_1, L_4 = \Box$, $L_2, L_3, L_5, L_6, L_7 = \epsilon$.
- If $L = \{a\Box\}$, $a \in A$, then $L_1, L_4, L_7 = \Box$, $L_2, L_3, L_5, L_6 = \epsilon$.
- If $L = L_1 \vee L_2$, then union of both rhs.
- If $L = L_1 + L_2$,

  $(L_1 + L_2) \cap B^* a B^* = (L_1 \cap B^* A B^* + L_2 \cap B^*) \vee (L_1 \cap B^* + L_2 \cap B^* A B^*)$
- If $L = L_1.L_2$,

  $(L_1.L_2) \cap B^* a B^* = ((L_1 \cap B^* A B^*).(L_2 \cap B^*)) \vee ((L_1 \cap B^*).(L_2 \cap B^* A B^*))$
- For the complementation, apart from the unambiguity, we need to strengthen the rhs slightly as follows.

  For each $a \in A$, we make L_1 a partition of contexts over B.

$$\begin{aligned} L_1(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \vee L'_1(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \\ = (L_1 \setminus L'_1)(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee (L'_1 \setminus L_1)(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \\ \vee (L_1 \cap L'_1)(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee (L_1 \cap L'_1)(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \end{aligned}$$

  Let L''_1 be the set of contexts which do not belong to any L_1. To make L_1's a partition we can add the product

$$L''_1(B^* + B^*(B^* + a\Box\emptyset + B^*) + B^*)$$

also to the rhs.

For any equivalence class L_1 from the above partition of L_1's, we can make L_2 also a partition:

$$\begin{aligned} L_1(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \vee L_1(L_2' + L_4'(L_5' + a\Box L_7' + L_6') + L_3') \\ = L_1((L_2 \setminus L'2) + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee L_1((L_2' \setminus L_2) + L_4'(L_5' + a\Box L_7' + L_6') + L_3') \\ \vee L_1((L_2 \cap L_2') + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee L_1((L_2 \cap L_2') + L_4'(L_5' + a\Box L_7' + L_6') + L_3') \end{aligned}$$

Moreover for L_2'', the set of contexts which do not belong to any L_2, we add the product

$$L_1(L_2'' + B^*(B^* + a\Box\emptyset + B^*) + B^*)$$

also to the rhs.

Similarly, for any given L_1 and L_2, we make L_3 a partition, and then L_4, L_5 and L_6 successively in that order. Once the rhs is in this form, we have :

$$(L_1)^c \cap B^*AB^* = \bigvee L_1(L_2 + L_4(L_5 + a\Box((L_7)^c) + L_6) + L_3)$$

This completes the proof of the splitting lemma.

□

D Splitting lemma for nested words

Please refer Figure 4 for a pictorial explanation of the terms.

Proof. The proof is by induction on the structure of the star free expression for L.

- If $L = \{a\}$ with $a \notin A$, or if $L = \emptyset$, or if $L = \{c\Box r\}$ with $c, r \notin A$ then $L_i = \emptyset$.
- If $L = \{a\}$, $a \in \mathfrak{I}, a \in A$, then Product (9) applies, with $L_5, L_6 = \epsilon$.
- If $L = \{c\Box r\}$ with $c \in A$ or $r \in A$, then Product (11) applies and $L_5, L_6 = \epsilon$
- If $L = L_1 \vee L_2$, then union of both rhs.
- If $L = L' \cdot L''$,

  $(L' \cdot L'') \cap B^*aB^* = ((L' \cap B^*AB^*) \cdot (L'' \cap B^*)) \vee ((L' \cap B^*) \cdot (L'' \cap B^*AB^*))$

- If $L = L' \boxdot L''$,

  $$(L' \boxdot L'') \cap B^* a B^* = ((L' \cap B^* A B^*) \boxdot (L'' \cap B^*)) \vee ((L' \cap B^*) \boxdot (L'' \cap B^* A B^*))$$

- For the complementation, apart from the unambiguity, we need to strengthen the rhs as before. We have more cases to consider depending on the form of the product ((1)...(11)).

  For each $c\Box r$ appearing in the rhs of the splitting lemma, and for each form in which it appear (not that while considering forms (1) and (4) (alternatively, (2) and(5) or (3) and (6)) are exactly the same for a given $c\Box r$), we make the L_i's a partition. Let us consider the form (1) (or (4)).

  $$\begin{aligned} L_1 \boxdot (L_2 \cdot L_4 \boxdot (L_5 \cdot c\Box r \boxdot L_7 \cdot L_6) \cdot L_3) \vee L'_1 \boxdot (L'_2 \cdot L'_4 \boxdot (L'_5 \cdot c\Box r \boxdot L'_7 \cdot L'_6) \cdot L'_3) \\ = (L_1 \setminus L'_1)(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee (L'_1 \setminus L_1)(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \\ \vee (L_1 \cap L'_1)(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee (L_1 \cap L'_1)(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \end{aligned}$$

  Let L''_1 be the set of contexts which do not belong to any L_1. To make L_1's a partition we can add the product

  $$L''_1(B^* + B^*(B^* + a\Box\emptyset + B^*) + B^*)$$

  also to the rhs.

  For any equivalence class L_1 from the above partition of L_1's, we can make L_2 also a partition:

  $$\begin{aligned} L_1(L_2 + L_4(L_5 + a\Box L_7 + L_6) + L_3) \vee L_1(L'_2 + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \\ = L_1((L_2 \setminus L'2) + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee L_1((L'_2 \setminus L_2) + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \\ \vee L_1((L_2 \cap L'_2) + L_4(L_5 + a\Box L_7 + L_6) + L_3) \\ \vee L_1((L_2 \cap L'_2) + L'_4(L'_5 + a\Box L'_7 + L'_6) + L'_3) \end{aligned}$$

  Moreover for L''_2, the set of contexts which do not belong to any L_2, we add the product

  $$L_1(L''_2 + B^*(B^* + a\Box\emptyset + B^*) + B^*)$$

  also to the rhs.

Similarly, for any given L_1 and L_2, we make L_3 a partition, and then L_4, L_5 and L_6 successively in that order. Once the rhs is in this form, we have :

$$(L_1)^c \cap B^*AB^* = \bigvee L_1(L_2 + L_4(L_5 + a\Box((L_7)^c) + L_6) + L_3)$$

This completes the proof of the splitting lemma.

□