Positively Approximating LTL Properties over Rewrite Rules

Pierre-Cyrille Héam[1] [(a)], Vincent Hugot[2] [(b)], Olga Kouchnarenko[1]

(1) FEMTO-ST CNRS 6174, Bourgogne Franche-Comté Univ. & Inria/CASSIS, France
(2) LIFL CNRS 8022 & Inria/Links, France

pheam@femto-st.fr vincent.hugot@inria.fr olga.kouchnarenko@inria.fr

Abstract. In formal verification, reachability analysis can be guided by temporal logic properties, for instance to achieve the counter-example-guided abstraction refinement (CEGAR) objectives. A way to perform this analysis is to translate a temporal logic formula expressed on maximal rewriting words into a "rewrite proposition" – a propositional formula whose atoms are language comparisons – and then to generate positive approximation procedures based on (approximations of) the rewrite proposition.

This paper provides a theoretical framework for both sub-problems on a fragment of LTL useful for safety properties, thus laying the foundation for a complete, automatic model-checking chain based on this method. For the first sub-problem, a set of translation rules yields the rewrite proposition; for the second, a set of procedure generation rules produces all possible theorems of the form "given some assumptions about the input system, this procedure decides (resp. positively approximates) whether the system satisfies the property".

Keywords: Linear Temporal Logic, Rewrite Systems, Rewrite Propositions, Regular Model-Checking

1. Introduction & Context

Term rewriting and rewriting logic have been intensively and successfully used for solving equational problems in automated deduction, for programming language definitions, for model transformations and generation of efficient interpreters as well as for specification and verification in software engineering. In this last context, system states are modelled by languages, while rewrite rules stand for *actions* of the system; for instance procedure or method calls. This technique has been successfully used to prove the security of cryptographic protocols [1] and Java Bytecode programs [2]. When proving security or safety, reachability analysis over sets of terms can be guided by temporal logic properties, for instance in [3, 4].

In particular, [4], which the present paper [(c)] aims to develop and complete, outlined a novel method for

[(a)] Partially supported by ANR 2010 BLAN 0202 02 FREC
[(b)] Partially supported by the French DGA (Direction Générale de l'Armement); this work was done while in FEMTO-ST.
Correspondence and offprint requests to: Héam, Hugot, Kouchnarenko
[(c)] This paper is an extended version of [5], and a slightly condensed version of the relevant parts of the thesis [6].

checking that the rules of a term-rewriting system (TRS, or rewrite system) are activated in accordance with a temporal property. Specifically, three Linear Temporal Logic (LTL) formulæ [(d)] were translated into equivalent rewrite propositions, which in turn were converted into algorithms that yield *yes* or *maybe*, which we call positive approximation procedures – the general problem is undecidable. For instance, given an initial language Π, a rewrite system $\mathcal{R}$ and two subsets $X, Y \subseteq \mathcal{R}$, the LTL property $\square(X \Rightarrow \bullet Y)$ – one of the three dealt with in [4] – signifies that whenever an accessible term is rewritten by some rewrite rule in X, the resulting term can be rewritten by some rule in Y, and not by any other rule. More concretely, if $\mathcal{R}$ models a cash machine, and $X = \{\,\texttt{ask\_PIN}\,\}$ and $Y = \{\,\texttt{auth}_1, \texttt{auth}_2, \texttt{cancel}\,\}$, then this property can be read as "whenever the user enters his or her PIN, then something happens immediately after, and that can only be either the authentication of the user (through either method 1 or 2) or the cancellation of the transaction —this excludes other possible but undesirable actions, such as sending the PIN over the network." Note that `ask_PIN` etcetera are, in this context, rewrite rules on trees representing the states of the machine. That property was shown to be satisfied if and only if the following rewrite proposition holds: $[\mathcal{R} \setminus Y]\,(X\,(\mathcal{R}^*(\Pi))) = \varnothing \;\wedge\; X\,(\mathcal{R}^*(\Pi)) \subseteq Y^{-1}(\mathcal{T})$, where $\mathcal{R}^*(\Pi)$ is the transitive and reflexive forward closure of Π by $\mathcal{R}$, and $\mathcal{T}$ is the set of all trees. The point of translating satisfaction in terms of rewrite propositions is that they present a more tractable intermediary form which can itself be translated into automata-based decision or positive approximation procedures. Indeed, if the initial language Π is regular, then the literature is rife with constructive results concerning questions such as preservation of regularity under forward closure, that is to say, "under which conditions on the rewrite system $\mathcal{R}$ is $\mathcal{R}^*(\Pi)$ still regular?". A recent and complete survey of such results can be found in [8, Sec. 2.1.1]. When preserving regularity is not an option, one may fall back on more expressive classes of tree automata (TA) such as Tree Automata with Global Equality constraints ($\mathsf{TA}^=$) [9]. As an example of both aspects, [4, Prop. 5] states that a language given by $\mathcal{R}^{-1}(\mathcal{T})$ can in all generality be represented by a $\mathsf{TA}^=$; furthermore, if $\mathcal{R}$ is left-linear, then the language is regular. Such results can be supplemented by regular approximation techniques; if $\mathcal{A}$ is a TA, such a procedure $\texttt{Approx}(\mathcal{A}, \mathcal{R})$ yields another TA $\mathcal{B}$ such that $\mathcal{L}(\mathcal{B}) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$, where $\mathcal{L}(\mathcal{A})$ is the language accepted by $\mathcal{A}$ [10, 11]. Current methods yield approximations that are tight enough to be quite useful. Put together, those tools provide a framework for building decision and positive approximation procedures from rewrite propositions. For instance, the proposition given above was shown to be positively approximated by the conjunction of the procedures $\texttt{IsEmpty}(\texttt{OneStep}(\mathcal{R} \setminus Y, \texttt{Approx}(\mathcal{A}, \mathcal{R})), X)$ and $\texttt{Subset}(\texttt{OneStep}(X, \texttt{Approx}(\mathcal{A}, \mathcal{R})), \texttt{Backward}(Y))$, where $\mathcal{L}(\mathcal{A}) = \Pi$ and under the additional constraint that Y must be left-linear. Note that, syntax notwithstanding, this is almost a straightforward reformulation of the original rewrite proposition.

To summarise the above, our approach to model-checking temporal properties of sequences of rewrite rules, as illustrated in [4], consists of two phases: **(1)** translation of a temporal logic formula expressed on maximal rewriting words into a rewrite proposition, which is independent of the specific properties of the system, and **(2)** translation of the rewrite proposition into positive approximation procedures, instantiating languages by means of automata, and introducing sound approximations where needed. In order for this approach to become useful for program verification, both steps must be generalised to arbitrary temporal properties – within an appropriate fragment of LTL – and automated. In the present paper, we develop automated methods for each phase on a fragment sufficient to express useful properties, and point out both limitations of the method and possible optimisations whenever relevant.

Note to reviewers:

This paper extends the previous IJCAR paper [5] in the following ways:

- A single translation rule ($X_\hbar$) – which is where most of the information stored during the translation is actually used – replaces and extends the previous method (which involved several, more complicated, less general rules). e.g. Sec. 3.1[p5].
- Sec. 5[p20], showing how to instantiate approximated algorithms from rewrite propositions, is new.
- Sec. 6[p24], presenting full derivations and discussing applicability, is new.
- Full proofs are given for most propositions – though, for want of space, some of them redirect to the thesis [6].
- There are far more examples, explanations, and intuitions interspersed throughout the text.

None of this material has been published in a workshop, conference, or journal, though a more discursive

(d) … chosen for their relevance to model-checking [7].

version of it exists in the relevant parts of the Ph.D. thesis [6], which is that of the second author (alphabetical order), the other authors being the thesis advisors.

Related work. In recent years, new results in rewriting logic have deeply extended the spectrum of its applications [12, 13, 14, 15], especially in relation with temporal logic for rewriting [16, 17]. Unlike [17], where LTL model checking is performed over finite structures, our approach handles temporal formulæ over infinite state systems. In this sense, it is close to [12]. However, in spite of its simplicity for practical applications, it does not permit – in its current state, at least – to consider equational theories. Our viewpoint differs slightly from other regular model-checking approaches such as Regular LTL [3] in that the temporal property relates to sequences of actions as opposed to sequences of states. It is however very similar to the method presented in [18], when reducing the equational theory to the identity. This work inserts itself within tree regular model-checking (TRMC), where it complements reachability analysis performed given a TRS model of transitions, as in [1, 2, 19, 20]. This adds to the already rich ecosystem surrounding TRS as a central formal model amenable to both execution and verification though model-checking, abstract interpretation, static analysis, interactive proofs etcetera [21, 10, 22, 23, 24].

Organisation of the paper. Section 2 presents the notions and notations in use throughout this paper, including the choice of temporal semantics and precise statements of the problems at hand. Section 3[p5] provides a brief intuitive introduction to our proposal for **(1)**, and develops the notion of *signatures* used in its resolution. Section 4[p13] relies on signatures to provide a set of translation rules that addresses **(1)**. Section 5[p20] focuses on **(2)**, which is addressed by means of procedure generation rules generating all possible positive approximation theorems. Section 6[p24] shows full derivations of both steps on toy formulæ, and examines whether and how well the developed methods apply to common temporal properties and existing TRS models in various domains.

2. Preliminaries & Problem Statement

The *extended naturals* are denoted by $\overline{\mathbb{N}} \triangleq \mathbb{N} \cup \{+\infty\}$. For any $k \in \mathbb{N}$, we let $\mathbb{N}_k \triangleq [k, +\infty) \cap \mathbb{N}$ and $\overline{\mathbb{N}}_k \triangleq \mathbb{N}_k \cup \{+\infty\}$. We write $[\![n, m]\!]$, where $n, m \in \mathbb{N}$, the integer interval $[n, m] \cap \mathbb{N}$, with the convention that $[\![n, +\infty]\!] \triangleq \mathbb{N}_n$. The powerset of S is written $\wp(S)$. Substitution is written $f[v/X]$, meaning "v replaces X in the expression f".

2.1. Rewrite Words & Maximal Rewrite Words

A comprehensive survey on term rewriting can be found in [25]. Let $\mathbb{A}$ be a ranked alphabet, $\mathcal{T}(\mathbb{A})$ (or simply $\mathcal{T}$) the set of all terms on $\mathbb{A}$, $\mathcal{R}$ a finite rewrite system, and $\Pi \subseteq \mathcal{T}$ any set of terms. A *finite or infinite word on* $\mathcal{R}$ is an element of

$$\mathcal{W} \triangleq \bigcup_{n \in \overline{\mathbb{N}}} \left([\![1, n]\!] \to \mathcal{R} \right) .$$

The length $\#w \in \overline{\mathbb{N}}$ of a word w is defined as $\mathrm{Card}\,(\mathrm{dom}\, w)$. Note that the empty function – of graph $\varnothing \times \mathcal{R} = \varnothing$ – is a word, which we call the *empty word*, denoted by λ. Let $w \in \mathcal{W}$ be a word of domain $[\![1, n]\!]$, for $n \in \overline{\mathbb{N}}$, and let $m \in \mathbb{N}_1$; then the *m-suffix of* w is the word denoted by w^m, such that

$$w^m \triangleq \left| \begin{array}{ccc} [\![1, n-m+1]\!] & \longrightarrow & \mathcal{R} \\ k & \longmapsto & w(k+m-1) \end{array} \right. .$$

Note that $w^1 = w$, for any word w. The intuitive meaning attached to a word w is a sequence of rewrite rules of $\mathcal{R}$representing a "run" of the TRS $\mathcal{R}$. Such a notion only makes sense with respect to initial terms to be rewritten. Thus we now define the *maximal rewrite words of $\mathcal{R}$, originating in* Π:

$$(\!|\Pi|\!) \triangleq \left\{ w \in \mathcal{W} \;\middle|\; \begin{array}{l} \exists u_0 \in \Pi\,; \exists u_1, \ldots, u_{\#w} \in \mathcal{T} : \forall k \in \mathrm{dom}\, w, \\ u_{k-1} \xrightarrow{w(k)} u_k \;\wedge\; \#w \in \mathbb{N} \Rightarrow \mathcal{R}(\{u_{\#w}\}) = \varnothing \end{array} \right\} .$$

Note the potential presence of the empty word in that set. Informally, a word w is in $(\!|\Pi|\!)$ if and only if the rewrite rules $w(1), \ldots, w(n), \ldots$ can be activated in succession, starting from a term $u_0 \in \Pi$, and the word w is "maximal" in the sense that it cannot be extended. Thus $(\!|\Pi|\!)$ captures the behaviours (or runs) of $\mathcal{R}$,

$$
\begin{array}{lll}
(w,i) \models X & \text{iff} & i \in \operatorname{dom} w \text{ and } w(i) \in X \\
(w,i) \models \neg\varphi & \text{iff} & (w,i) \not\models \varphi \\
(w,i) \models (\varphi \wedge \psi) & \text{iff} & (w,i) \models \varphi \text{ and } (w,i) \models \psi \\
(w,i) \models \bullet^m \varphi & \text{iff} & i+m \in \operatorname{dom} w \text{ and } (w,i+m) \models \varphi \\
(w,i) \models \circ^m \varphi & \text{iff} & i+m \notin \operatorname{dom} w \text{ or } (w,i+m) \models \varphi \\
(w,i) \models \varphi \,\mathbf{U}\, \psi & \text{iff} & \exists j \in \operatorname{dom} w : j \geqslant i \wedge \left\{ \begin{array}{l} (w,j) \models \psi \quad \wedge \\ \forall k \in [\![i, j-1]\!],\ (w,k) \models \varphi \end{array} \right. \\
\\
(w,i) \models \top & & (w,i) \not\models \bot \\
(w,i) \models \neg X & \text{iff} & i \notin \operatorname{dom} w \text{ or } w(i) \notin X \\
(w,i) \models (\varphi \vee \psi) & \text{iff} & (w,i) \models \varphi \text{ or } (w,i) \models \psi \\
(w,i) \models (\varphi \Rightarrow \psi) & \text{iff} & (w,i) \models \varphi \Rightarrow (w,i) \models \psi \\
(w,i) \models \Diamond \varphi & \text{iff} & \exists j \in \operatorname{dom} w : j \geqslant i \wedge (w,j) \models \varphi \\
(w,i) \models \Box \varphi & \text{iff} & \forall j \in \operatorname{dom} w,\ j \geqslant i \Rightarrow (w,j) \models \varphi
\end{array}
$$

For any $w \in \mathcal{W}$, $i \in \mathbb{N}_1$, $m \in \mathbb{N}$ and $X \in \wp(\mathcal{R})$.

Fig. 1: LTL Semantics on Maximal Rewrite Words

starting from Π; this notion is equivalent the full paths of the rewrite graph described in [4], and corresponds to the usual maximal trace semantics [26], with a focus on transitions instead of states.

2.2. Defining Temporal Semantics on Rewrite Words

Choice of LTL & Syntax. Let us define precisely the kind of temporal formulæ under consideration, and their semantics. We shall use a slight variant of LTL with generalised weak and strong next operators; the reasons for this choice will be discussed when the semantics are examined. A formula $\varphi \in \text{LTL}$ is generated by the following grammar:

$$
\begin{array}{ll}
\varphi := X \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bullet^m \varphi \mid \circ^m \varphi \mid \varphi \,\mathbf{U}\, \varphi & X \in \wp(\mathcal{R}) \\
\qquad \top \mid \bot \mid \varphi \vee \varphi \mid \varphi \Rightarrow \varphi \mid \Diamond \varphi \mid \Box \varphi & m \in \mathbb{N}\,.
\end{array}
$$

Note that the operators which appear on the first line are functionally complete; the remaining operators are defined syntactically as: $\top \triangleq \mathcal{R} \vee \neg\mathcal{R}$, $\bot \triangleq \neg\top$, $\varphi \vee \psi \triangleq \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \Rightarrow \psi \triangleq \neg\varphi \vee \psi$, $\Diamond \varphi \triangleq \top \,\mathbf{U}\, \varphi$ and $\Box \varphi \triangleq \neg \Diamond \neg\varphi$.

Choice of Semantics. In the literature, the semantics of LTL are defined and well-understood for ω-words; however the words of $(\!|\Pi|\!)$ may be finite – even empty – or infinite, which corresponds to the fact that, depending on its input, a rewrite system may either not terminate, terminate after some rewrite operations, or terminate immediately. Therefore we need semantics capable of accommodating both ω-words and finite words, including the edge-case of the empty word. Figure 1[p4] presents our choice of semantics for this paper, which is essentially Finite-LTL [F-LTL, 7] with generalised next operators and the added twist that words may be infinite or empty. Note that $\bullet^1$ and $\circ^1$ correspond exactly to the classical strong and weak next operators, and that for $m \geqslant 1$, $\bullet^m$ (resp. $\circ^m$) can trivially be obtained by repeating $\bullet^1$ (resp. $\circ^1$) m times. So the only non-trivial difference here is the existence of $\bullet^0$ and $\circ^0$; this will prove quite convenient when we deal with the translation of $\Box$, using the following lemma.

Lemma 1 *(Weak-Next & Always). Let $\varphi \in \text{LTL}$, $w \in \mathcal{W}$, $k \in \mathbb{N}$ and $i \in \mathbb{N}_1$; it holds that* **(1)** *$(w,i) \models \Box\varphi$ iff $(w,i) \models \bigwedge_{m=0}^{\infty} \circ^m \varphi$ and* **(2)** *$(w,i) \models \Box\varphi$ iff $(w,i) \models \bigwedge_{m=0}^{k-1} (\circ^m \varphi) \wedge \circ^k \Box\varphi$.*

Short Proof. **(1)** $(w,i) \models \bigwedge_{m=0}^{\infty} \circ^m \varphi \iff \bigwedge_{m=0}^{\infty} (w,i) \models \circ^m \varphi \iff \forall j \in \operatorname{dom} w,\ j \geqslant i \Rightarrow (w,j) \models \varphi \iff (w,i) \models \Box\varphi$. **(2)** $(w,i) \models \bigwedge_{m=0}^{\infty} \circ^m \varphi \iff (w,i) \models \bigwedge_{m=0}^{k-1} (\circ^m \varphi) \ \wedge\ \bigwedge_{m=k}^{\infty} (\circ^m \varphi)\, (w, i+k) \models \Box\varphi \iff (w,i) \models \bigwedge_{m=0}^{k-1} (\circ^m \varphi) \ \wedge\ \circ^k \Box\varphi$. □

Before moving on, let us stress that the choice of LTL must be considered as data for the purposes of this paper.

TRS & LTL. Let φ be an LTL formula. It is said that a word w satisfies/is a model of φ (denoted by

$w \models \varphi$) iff $(w, 1) \models \varphi$. Alternatively, we have $(w, i) \models \varphi$ iff $w^i \models \varphi$. We say that the rewrite system $\mathcal{R}$, with initial language Π, satisfies/is a model of φ (denoted by $\mathcal{R}, \Pi \models \varphi$) iff $\forall w \in (\!|\Pi|\!),\ w \models \varphi$.

2.3. Rewrite Propositions & Problem Statement

A *rewrite proposition on $\mathcal{R}$, from* Π is a formula of propositional logic whose atoms are language or rewrite systems comparisons. More specifically, a rewrite proposition π is generated by the following grammar:

$$\pi := \gamma \mid \gamma \wedge \gamma \mid \gamma \vee \gamma \qquad \gamma := \ell = \varnothing \mid \ell \subseteq \ell \qquad X \in \wp(\mathcal{R})\ .$$
$$\ell := \Pi \mid \mathcal{T} \mid X(\ell) \mid X^{-1}(\ell) \mid X^*(\ell)$$

Since the comparisons γ have obvious truth values, the interpretation of rewrite propositions is trivial; thus π is automatically confused with its truth value in the remainder of this paper.

Problem Statements. The overarching goal is a systematic method to positively approximate whether $\mathcal{R}, \Pi \models \varphi$, given a rewrite system $\mathcal{R}$, a temporal formula φ in LTL, and an initial language $\Pi \subseteq \mathcal{T}$. This goal is broken down into two distinct sub-problems:

(1) Finding an algorithmic method for building, from φ, a rewrite proposition π such that $\mathcal{R}, \Pi \models \varphi$ if and only if π holds. We call such a method, as well as its result, an *exact translation* of φ, and say that π translates φ.

(2) Finding an algorithm for generating, from π, a positive approximation procedure δ, that answers positively only if π holds, or a full decision procedure, whenever possible.

By solving both sub-problems, one has $\delta \implies \pi$ and $\pi \iff \mathcal{R}, \Pi \models \varphi$, and therefore $\delta \implies \mathcal{R}, \Pi \models \varphi$, which achieves the overall goal.

One notices that the full equivalence is not needed in $\pi \Leftrightarrow \mathcal{R}, \Pi \models \varphi$; if π is only a sufficient (resp. necessary) condition, then it is an *under-approximated* (resp. *over-approximated*) translation. Of course, only under-approximated translations hold any practical interest for our purposes. Regarding the first problem, this paper concerns itself only with exact translations.

3. Technical Groundwork: Antecedent Signatures

The first problem is tackled by two complementary tools: *signatures*, developed in this section, and *translation rules* in Sec. 4[p13] relying on signatures.

3.1. Overview & Intuitions

The Base Cases. Counterintuitively, $\varphi = \neg X$ is actually a simpler case than $\varphi = X$ as far as the translation is concerned, so it will be considered first.

Case 1: Negative Literal. Suppose $\mathcal{R}, \Pi \models \neg X$. Recalling the semantics in Fig. 1[p4], this means that no term of Π can be rewritten by a rule in X. They may or may not be rewritable by rules not in X, though. Consider now

$$\pi_1 \equiv X(\Pi) = \varnothing\ ; \tag{π_1}$$

it is easy to become convinced that this is an exact translation.

Case 2: Positive Literal. Let $\varphi = X$. Writing $\pi_2 \equiv [\mathcal{R} \setminus X](\Pi) = \varnothing$ translates the fact that only rules of X can rewrite Π, but there is nothing in π_2 to enforce that. Looking at the semantics, all possible words of $(\!|\Pi|\!)$ must have at least one move (i.e. $1 \in \operatorname{dom} w$). It means that all terms of Π are rewritable: $\Pi \subseteq \mathcal{R}^{-1}(\mathcal{T})$. More specifically, since we already impose that they are not rewritable by $\mathcal{R} \setminus X$, they are rewritable by X, i.e. $\Pi \subseteq X^{-1}(\mathcal{T})$. We obtain an exact translation:

$$\pi_2' \equiv [\mathcal{R} \setminus X](\Pi) = \varnothing\ \wedge\ \Pi \subseteq X^{-1}(\mathcal{T})\ . \tag{π_2'}$$

Of Strength & Weakness. Let us reflect on the previous cases; the immediate intuition is that X is

stronger than $\neg X$, in the sense that whenever one sees X, one must write an additional clause – enforcing rewritability – compared to $\neg X$. This actually depends on the context, as the next example will show.

Case 3: Always Negative. Let $\varphi = \Box\,\neg X$. This means that neither the terms of Π nor their successors can be rewritten by X; in other words $\pi_3 \equiv X\,(\mathcal{R}^*(\Pi)) = \varnothing$. The translation is almost the same as for $\neg X$ in π_1. More formally,

$$\pi_3 \equiv X\,(\mathcal{R}^*(\Pi)) = \varnothing \equiv \pi_1[\mathcal{R}^*(\Pi)/\Pi]\;. \tag{π_3}$$

Case 4: Always Positive. In general, under the semantics for $\Box$, words of any length may satisfy $\Box\,X$. Differently from the relationship between the translations of $\neg X$ and $\Box\,\neg X$, the correct translation is simply

$$\pi'_4 \equiv [\mathcal{R} \setminus X]\,(\mathcal{R}^*(\Pi)) = \varnothing\;. \tag{π'_4}$$

Unlike Cases 1 and 2, X is not in any sense stronger than $\neg X$ when behind a $\Box$. This is an important point which we shall need to keep track of during the translation; that necessary bookkeeping will be done by means of the *signatures* introduced in Sec. 3.3[p7].

Conjunction, Disjunction & Negation. Case 5: And & Or. It is pretty clear that if π_5 translates φ and π'_5 translates ψ, then $\pi_5 \wedge \pi'_5$ translates $\varphi \wedge \psi$. This holds thanks to the implicit universal quantifier, as we have $(\mathcal{R}, \Pi \models \varphi \wedge \psi) \iff (\mathcal{R}, \Pi \models \varphi) \wedge (\mathcal{R}, \Pi \models \psi)$. Contrariwise, the same does not hold for the disjunction, and we have no general solution [(e)] to handle it. Given that one of the implications still holds, namely $(\mathcal{R}, \Pi \models \varphi \vee \psi) \impliedby (\mathcal{R}, \Pi \models \varphi) \vee (\mathcal{R}, \Pi \models \psi)$, a crude under-approximation can still be given if all else fails:

$$\pi_5 \vee \pi'_5 \implies \mathcal{R}, \Pi \models \varphi \vee \psi\;. \tag{π''_5}$$

Case 6: Negation. Although we have seen in Case 1 that a negative literal can easily be translated, negation cannot be handled in all generality by our method. Note that, because of the universal quantification, $\mathcal{R}, \Pi \not\models \varphi \neq \mathcal{R}, \Pi \models \neg\varphi$; thus the fact that π_6 translates φ does not a priori imply that $\neg\pi_6$ translates $\neg\varphi$. This is why we shall assume in practice that input formulæ are provided in a sanitised form, where negations only appear on literals. The presence of both weak and strong next operators facilitates this, as $\neg \circ^m \varphi \Leftrightarrow \bullet^m \neg\varphi$. Note that this is not exactly the same as requiring a Negative Normal Form (NNF).

Handling Material Implication. Case 7: Implication. Inasmuch as $\varphi \Rightarrow \psi$ is defined as $\neg\varphi \vee \psi$ (Cases 5 and 6), must material implication be forgone as well? An example involving an implication has been given in the introduction (page 1), so it would seem that a translation can be provided in at least some cases. Let us take the simple example $X \Rightarrow \bullet Y$. Assuming that any term $u \in \Pi$ is rewritten into some u' by a rule in X, then u' must be rewritable by Y, and only by Y. The set of X-successors of Π being $X(\Pi)$, those conditions yield the translation

$$\pi_7 \equiv X(\Pi) \subseteq Y^{-1}\,(\mathcal{T}) \;\wedge\; [\mathcal{R} \setminus Y]\,(X(\Pi)) = \varnothing\;. \tag{π_7}$$

Note that the way in which implication has been handled here is very different from the approach taken for the other binary operators: the antecedent of the implication was "assumed", whilst the consequent was translated as usual. In fact, recalling that π'_2 translates X, and thus $\pi''_2 \equiv \pi'_2[Y/X]$ translates Y, we have $\pi_7 \equiv \pi''_2[X(\Pi)/\Pi]$. So, "assuming" the antecedent consisted simply in changing our set of reachable terms, called the *past* from now on, hence the notation Π. When considering $\bullet Y \Rightarrow X$, the antecedent lies in the future, relatively to the consequent. Therefore, in order to deal with all cases, Section 3.3[p7] introduces the *signatures*.

3.2. Choosing a Suitable Fragment of LTL

As negation of complex formulæ is problematic, we shall therefore work only with formulæ in a sanitised form, such that negation appears only on literals, and implication remains allowed. For instance, $(A \vee B) \Rightarrow C$, $(A \Rightarrow C) \wedge (B \Rightarrow C)$, and $(\neg A \wedge \neg B) \vee C$ are three equivalent formulæ, all sanitised. However, the first and second forms will be favoured over the third (the NNF), because those forms fit into the translation system presented hereafter. Furthermore, there are operators – such as $\Diamond$ – for which it seems that no exact translation can be provided, since rewrite propositions are not expressive enough; in particular, $\mathcal{R}^*(\Pi)$

(e) There are however special cases where disjunction can be translated exactly; see rules $(\vee^{\Rightarrow}_{\wedge})$[p14] and $(\vee^{\neg}_{\Rightarrow})$.

hides all information regarding finite or infinite traces. Hence, none of the operators of the "Until" family $\{\, \Diamond, \mathbf{U}, \mathbf{W}, \mathbf{R}, \dots \}$ can be translated exactly, As this paper focusses on exact translations, we shall work chiefly within the following fragment of LTL, denoted by $\mathcal{R}$-LTL:

$$\begin{array}{ll} \varphi := X \mid \neg X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \Rightarrow \varphi \mid & X \in \wp(\mathcal{R}) \\ \quad \bullet^m \varphi \mid \circ^m \varphi \mid \Box \varphi & m \in \mathbb{N}\ . \end{array}$$

3.3. Girdling the Future: Signatures

To deal with implication (cf. Sec. 3.1[p5], Case 7), our approach consists in building a model of φ – called a *signature* of φ, written $\xi(\varphi)$. This model will also be used to store sufficient information regarding the context in order to determine whether the translation ought to be "strong" or "weak".

The variety of signatures defined hereafter handles formulæ φ within the fragment $\mathcal{A}$-LTL ($\mathcal{A}$ for **a**ntecedent), which is $\mathcal{R}$-LTL without $\vee$ or $\Rightarrow$. This section defines signatures formally and presents a suitable map $\xi(\cdot) : \mathcal{A}\text{-LTL} \to \Sigma$, where Σ is the space of signatures, whose correctness proof is broken down in eight main lemmata, and finally summarised by Thm. 15[p13].

The *informal* idea behind our definition of signatures is to capture information regarding the possible successive rewriting steps from the current language Π. The empty signature encodes no information at all: Starting from $t \in \Pi$, there may or may not be a rewriting transition $t \xrightarrow{r} t'$ and even if there is all that can be said at this point is that $r \in \mathcal{R}$; moreover, no further information is available about t' and its possible successors. But as more information is gained from antecedents, constraints will be added to the signature. The compatibility of a rewrite word with a signature gives them precise semantics; it will be made explicit by the definition of *constrained words* below.

Signatures. A *signature* σ is an element of the space

$$\Sigma = \bigcup_{n \in \mathbb{N}} \Big[\big([\![1, n]\!] \cup \{\omega\} \big) \to \wp(\mathcal{R}) \Big] \times \wp(\overline{\mathbb{N}})\ .$$

Core, Support, Domain, Cardinal. Let $\sigma = (f, S)$; then the function f is called the *core of* σ, denoted by $\partial\sigma$, and S is called its *support*, written $\nabla\sigma$. The *domain of* σ is defined as $\operatorname{dom}\sigma \triangleq \operatorname{dom} f \setminus \{\omega\}$, and its *cardinal* is $\#\sigma \triangleq \operatorname{Card}(\operatorname{dom}\sigma)$.

Special Notations, Empty Signature. A signature $\sigma = (f, S)$ will be written either compactly as $\sigma = \wr f \mid S \int$, or *in extenso* as

$$\wr f(1), f(2), \dots, f(\#\sigma) \fatsemi f(\omega) \mid S \int\ .$$

We denote by $\varepsilon \triangleq \wr \fatsemi \mathcal{R} \mid \overline{\mathbb{N}} \int$ the *empty signature*. Let $k \in \mathbb{N}_1 \cup \{\omega\}$, then we write

$$\sigma[k] \triangleq \begin{cases} f(k) & \text{if } k \in \operatorname{dom}\sigma \\ f(\omega) & \text{if } k \notin \operatorname{dom}\sigma \end{cases}\ .$$

The notation $\sigma[k]$ is read "σ at (position) k" and is referred to as the *at operator*.

Constrained Words. The set of maximal rewrite words which satisfy the constraints encoded by a signature assigns precise semantics to the signatures we build. The *maximal rewrite words of* $\mathcal{R}$*, originating in* Π *and constrained by* σ are defined by

$$(\!|\Pi \fatsemi \sigma|\!) \triangleq \{\, w \in (\!|\Pi|\!) \mid \#w \in \nabla\sigma \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \sigma[k] \,\}\ .$$

For the sake of conciseness, we write $(\!|\Pi \fatsemi \sigma|\!)^{\#}_m$ the set $\{\, w \in (\!|\Pi \fatsemi \sigma|\!) \mid \#w = m \,\}$, $(\!|\Pi \fatsemi \sigma|\!)^{\#}_S$ the set $\{\, w \in (\!|\Pi \fatsemi \sigma|\!) \mid \#w \in S \,\}$, and $(\!|\Pi \fatsemi \sigma|\!)^{\#}_{\prec n}$ the set $\{\, w \in (\!|\Pi \fatsemi \sigma|\!) \mid \#w \prec n \,\}$, for $(\prec) \in \{\, <, >, \leqslant, \geqslant \,\}$ and $n \in \overline{\mathbb{N}}$.

Example: Let $\sigma = \wr X, Y \fatsemi Z \mid \mathbb{N}_2 \int$; then its core is the function $\partial\sigma = \{\ 1 \mapsto X, 2 \mapsto Y, \omega \mapsto Z\ \}$, its domain is $\operatorname{dom}\sigma = [\![1, 2]\!]$, its support is $\nabla\sigma = \mathbb{N}_2$, its cardinal is $\#\sigma = 2$, and we have $\sigma[1] = X$, $\sigma[2] = Y$, $\sigma[3] = \sigma[4] = \cdots = \sigma[\omega] = Z$. Its constrained words are the maximal rewrite words of length at least 2, whose first two letters are in X and Y, respectively, and whose other letters are all in Z. $\Diamond$

Our objective in this section is in particular to define a map

$$\xi(\cdot) : \mathcal{A}\text{-LTL} \longrightarrow \Sigma$$

such that $\xi(\varphi)$ is a model of φ, in the sense that the maximal rewrite words constrained by $\xi(\varphi)$ are exactly those which satisfy φ. In formal terms, we expect $\xi(\cdot)$ to satisfy the following property, for all $\Pi \subseteq \mathcal{T}$ and $\varphi \in \mathcal{A}$-LTL:

$$(\!|\Pi \wr \xi(\varphi)|\!) = \{\, w \in (\!|\Pi|\!) \mid w \models \varphi \,\} \ . \tag{3.1}$$

The map $\xi(\cdot)$ will have to be built inductively on the structure of its argument. Let us start by observing that the empty signature carries no constraint at all, which bridges constrained words and maximal rewrite words:

Lemma 2 (*No Constraints*)**.** *It holds that* $(\!|\Pi \wr \varepsilon|\!) = (\!|\Pi|\!)$.

Proof. For this first proof, all steps have been detailed. We have

$$\begin{aligned}
(\!|\Pi \wr \varepsilon|\!) &= (\!|\Pi \wr \wr \mathcal{R} \mid \overline{\mathbb{N}} \wr|\!) && \text{(i)}\\
&= \{\, w \in (\!|\Pi|\!) \mid \#w \in \nabla\varepsilon \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \varepsilon[k] \,\} && \text{(ii)}\\
&= \{\, w \in (\!|\Pi|\!) \mid \#w \in \overline{\mathbb{N}} \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \mathcal{R} \,\} && \text{(iii)}\\
&= \{\, w \in (\!|\Pi|\!) \mid \top \ \wedge\ \top \,\} = \{\, w \in (\!|\Pi|\!) \,\} = (\!|\Pi|\!) \ , && \text{(iv)}
\end{aligned}$$

where step (i) proceeds by definition of the empty signature, step (ii) by definition of constrained rewrite words, step (iii) by definition of the "at operator" for signatures, and step (iv) rests on all lengths being in $\overline{\mathbb{N}}$, and all rules in $\mathcal{R}$. □

As an immediate consequence of the above, the empty signature is a model of $\top$.

Lemma 3 (*True*)**.** *Taking* $\xi(\top) \triangleq \varepsilon$ *satisfies* (3.1).

Proof. $(\!|\Pi \wr \xi(\top)|\!) = (\!|\Pi \wr \varepsilon|\!) = (\!|\Pi|\!) = \{\, w \in (\!|\Pi|\!) \mid w \models \top \,\}$. □

Conversely, to handle $\bot$, we need a signature that rejects every possible rewrite word; in that case there are many possible, equally valid choices, the most straightforward of which is as follows:

Lemma 4 (*False*)**.** *Taking* $\xi(\bot) \triangleq \wr \varnothing \mid \varnothing \wr$ *satisfies* (3.1).

Proof. The result rests on $x \in \varnothing$ being false for all x. □

The next lemma claims that for positive literals the proposed translation was correct.

Lemma 5 (*Positive Literal*)**.** *Taking* $\xi(X) \triangleq \wr X \wr \mathcal{R} \mid \overline{\mathbb{N}}_1 \wr$ *satisfies* (3.1).

Proof. Straightforward by translating $\#w$ in terms of $\operatorname{dom} w$. □

Negative literals are translated in roughly the same way as positive ones, the main difference being that the length 0 is not excluded from their support.

Lemma 6 (*Negative Literal*)**.** *Taking* $\xi(\neg X) \triangleq \wr \mathcal{R} \setminus X \wr \mathcal{R} \mid \overline{\mathbb{N}} \wr$ *satisfies* (3.1).

Proof. It suffices to break up the $\forall k \in \operatorname{dom} w$. □

Now that the base cases are all covered, we move on to the inductive cases, the first of which is conjunction. Let us take the simplest possible example: $X \wedge Y$. We have by Lemma 5

$$\xi(X) = \wr X \wr \mathcal{R} \mid \overline{\mathbb{N}}_1 \wr \quad \text{and} \quad \xi(Y) = \wr Y \wr \mathcal{R} \mid \overline{\mathbb{N}}_1 \wr \ ,$$

but also, considering that $X \cap Y$ is a positive literal as well:

$$\xi(X \cap Y) = \wr X \cap Y \wr \mathcal{R} \mid \overline{\mathbb{N}}_1 \wr \ .$$

It should be intuitively pretty clear that $\xi(X \cap Y)$ encodes both the constraints of $\xi(X)$ and $\xi(Y)$. However, the general idea that "conjunction" between signatures is translated by intersections stands on its own, as the next lemmata will show.

Signature Product. Let σ and σ' be two signatures; then their *product* is another signature defined as $\sigma \otimes \sigma' \triangleq \wr g \mid \nabla\sigma \cap \nabla\sigma' \wr$, where

$$g \triangleq \left| \begin{array}{ccc} \operatorname{dom}\partial\sigma \cup \operatorname{dom}\partial\sigma' & \longrightarrow & \wp(\mathcal{R}) \\ k & \longmapsto & \sigma[k] \cap \sigma'[k] \end{array} \right. .$$

Note that as a consequence, $\forall k \in \mathbb{N}_1$, $(\sigma \otimes \sigma')[k] = \sigma[k] \cap \sigma'[k]$.

Example: Let us take the two signatures $\sigma = \wr X, Y \fatsemi Z \mid \mathbb{N}_2 \wr$ and $\rho = \wr X' \fatsemi Z' \mid \mathbb{N}_3 \wr$; then $\sigma \otimes \rho = \wr X \cap X', Y \cap Z' \fatsemi Z \cap Z' \mid \mathbb{N}_3 \wr$. ♢

Signature product is fairly well-behaved with respect to constrained words, and can be "broken down" and replaced by the intersection of simpler constrained sets. The next lemma and its generalisation (3.3)[p11] show this to be true of finite products, and this property will later be generalised to infinite products (Lem. 12[p11]).

Lemma 7 (*Breaking Finite Products*)**.** *For any signatures $\sigma, \rho \in \Sigma$, and any language Π, we have $(\!|\Pi \fatsemi \sigma \otimes \rho|\!) = (\!|\Pi \fatsemi \sigma|\!) \cap (\!|\Pi \fatsemi \rho|\!)$.*

Proof. $(\!|\Pi \fatsemi \sigma \otimes \rho|\!)$ is, by definition, the set of $w \in (\!|\Pi|\!)$ such that

$$\begin{array}{rl} & \#w \in \nabla\sigma \cap \nabla\rho \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \sigma[k] \cap \rho[k] \\ \iff & \#w \in \nabla\sigma \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \sigma[k] \\ \wedge & \#w \in \nabla\rho \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \rho[k] \\ \iff & w \in (\!|\Pi \fatsemi \sigma|\!) \ \wedge\ w \in (\!|\Pi \fatsemi \rho|\!) \\ \iff & w \in (\!|\Pi \fatsemi \sigma|\!) \cap (\!|\Pi \fatsemi \rho|\!)\ . \end{array}$$

Thus $(\!|\Pi \fatsemi \sigma \otimes \rho|\!) = (\!|\Pi|\!) \cap (\!|\Pi \fatsemi \sigma|\!) \cap (\!|\Pi \fatsemi \rho|\!) = (\!|\Pi \fatsemi \sigma|\!) \cap (\!|\Pi \fatsemi \rho|\!)$. □

Lemma 8 (*Conjunction*)**.** *Provided that the subformulæ $\xi(\varphi)$ and $\xi(\psi)$ satisfy (3.1), taking $\xi(\varphi \wedge \psi) \triangleq \xi(\varphi) \otimes \xi(\psi)$ satisfies (3.1).*

Proof. Straightforward using the previous lemma. □

We shall generalise results on signature products to infinitary cases later on, to encode $\Box\varphi$. Meanwhile, let us turn our attention to the strong and weak next operators. Let us consider a formula φ whose signature is given by

$$\xi(\varphi) = \wr W, X, Y \fatsemi Z \mid \overline{\mathbb{N}}_2 \wr\ ,$$

and propose plausible candidates for $\xi\big(\circ^1\varphi\big)$ and $\xi\big(\bullet^1\varphi\big)$. Although we still lack all the formal tools to prove it, after numerous similar examples it is easy to derive that

$$\varphi = W \wedge \bullet^1 X \wedge \circ^2 Y \wedge \circ^3 \Box\, Z\ .$$

The lengths of the words $w \models \circ^1\varphi$ can therefore be enumerated. By the semantics of $\circ^1$, if $\#w = 0$ or $\#w = 1$, then w is automatically a model of $\circ^1\varphi$. Suppose that $\#w \geqslant 2$; then it must be the case that $w^2 \models \varphi$. In particular, because of the strong next this means that $\#w^2 \in \overline{\mathbb{N}}_2$, in other words $\#w^2 \geqslant 2$; thus, since $\#w^2 = \#w - 1$, we have $\#w \geqslant 3$. So the set of acceptable lengths is $[\![0, 1]\!] \cup \overline{\mathbb{N}}_3 = \overline{\mathbb{N}} \setminus \{2\}$. Now, as far as the rewrite rules are concerned, the first rule, if it exists, can be anything; obviously the second one must live in X, the third in Y, and all the subsequent rules must come from Z. So we have derived the following signature:

$$\xi\big(\circ^1\varphi\big) = \wr \mathcal{R}, W, X, Y \fatsemi Z \mid [\![0, 1]\!] \cup \overline{\mathbb{N}}_3 \wr\ .$$

It is clear that $\xi\big(\bullet^1\varphi\big)$ will have the same core as $\xi\big(\circ^1\varphi\big)$, but a different support. Indeed in that case the length 0 and 1 are clearly not suitable, while the rest of the previous reasoning still holds. Thus we have immediately

$$\xi\big(\bullet^1\varphi\big) = \wr \mathcal{R}, W, X, Y \fatsemi Z \mid \overline{\mathbb{N}}_3 \wr\ .$$

The work done on this example is generalised in the next definitions, and this suffices to compute $\xi\big(\bullet^1\varphi\big)$ and $\xi\big(\circ^1\varphi\big)$ for all φ, as shown in the next lemma.

Arithmetic Overloading. We overload the operator $+$ on the profile $\wp(\overline{\mathbb{N}}) \times \mathbb{N} \to \wp(\overline{\mathbb{N}})$ such that, for

any $S \in \wp(\overline{\mathbb{N}})$ and $n \in \mathbb{N}$, we have

$$S + n \triangleq \{\, k + n \mid k \in S \,\} \ .$$

Right Shifts. Let $\sigma \in \Sigma$, $m \in \mathbb{N}$ and $\mathcal{R}_1 = \mathcal{R}$, $\ldots$, $\mathcal{R}_m = \mathcal{R}$; then we define the *weak m-right shift of σ* as

$$\sigma \rhd m \triangleq \wr \mathcal{R}_1, \ldots, \mathcal{R}_m, \partial\sigma(1), \ldots, \partial\sigma(\#\sigma) \fatsemi \partial\sigma(\omega) \mid [\![0, m]\!] \cup (\nabla\sigma + m) \wr \ ,$$

while the *strong m-right shift of σ* is

$$\sigma \blacktriangleright m \triangleq \wr \mathcal{R}_1, \ldots, \mathcal{R}_m, \partial\sigma(1), \ldots, \partial\sigma(\#\sigma) \fatsemi \partial\sigma(\omega) \mid (\nabla\sigma \setminus \{0\}) + m \wr \ .$$

The only point in those definitions that was not readily apparent in the above example is the $\nabla\sigma \setminus \{0\}$ appearing in the support of the right shift. The reason for its introduction is best understood in the context of a formula φ whose signature admits zero in its support; when behind a strong next of level zero ($\bullet^0$), the only change in the support of the signature is the removal of the zero. The proofs of Lem. 9, 10 and 13[p12] make use of immediate equations: for all $m \in \mathbb{N}$ and all $k \in \mathbb{N}_1$, if $k \leqslant m$, $(\sigma \blacktriangleright m)[k] = (\sigma \rhd m)[k] = \mathcal{R}$ and if $k > m$, $(\sigma \blacktriangleright m)[k] = (\sigma \rhd m)[k] = \sigma[k - m]$.

Lemma 9 (*Weak Next*). *Provided that the signature of the subformula $\xi(\varphi)$ satisfies* (3.1)*, taking $\xi(\circ^m\varphi) \triangleq \xi(\varphi) \rhd m$ satisfies* (3.1)*.*

Proof. We write $\sigma = \xi(\varphi)$ and $\sigma_m = \xi(\varphi) \rhd m$. Let $w \in (\!|\Pi|\!)$, then $w \models \circ^m\varphi$ iff

$$\begin{aligned}
& 1 + m \notin \operatorname{dom} w \ \vee \ w^{1+m} \models \varphi \iff \#w \in [\![0, m]\!] \ \vee \ w^{1+m} \models \varphi \\
\iff & \#w \in [\![0, m]\!] \ \vee \ (\#w^{1+m} \in \nabla\sigma \ \wedge \ \forall k \in \operatorname{dom} w^{1+m}, \ w^{1+m}(k) \in \sigma[k]) \\
\iff & \#w \in [\![0, m]\!] \ \vee \ (\#w - m \in \nabla\sigma \ \wedge \ \forall k \in [\![1 + m, \#w]\!] \, , \ w(k) \in \sigma[k - m]) \\
\iff & (\#w \in [\![0, m]\!] \ \vee \ \#w \in \nabla\sigma + m) \ \wedge \ (\forall k \in [\![1 + m, \#w]\!] \, , \ w(k) \in \sigma_m[k]) \\
\iff & \#w \in \nabla\sigma_m \ \wedge \ \forall k \in \operatorname{dom} w, \ w(k) \in \sigma_m[k] \iff w \in (\!|\Pi \fatsemi \sigma_m|\!) \ .
\end{aligned}$$

Note that $\#w^{1+m} = \#w - m$ only holds because we can safely assume, by the left member of the disjunction, that $\#w \in [\![0, m]\!]$, or in other words, $\#w > m$. □

Lemma 10 (*Strong Next*). *Provided that the signature of the subformula $\xi(\varphi)$ satisfies* (3.1)*, taking $\xi(\bullet^m\varphi) \triangleq \xi(\varphi) \blacktriangleright m$ satisfies* (3.1)*.*

A proof can be found in [6].

Now there only remains to deal with the last inductive case: the $\Box$ operator. To this end, we recall Lem. 1[p4]:

$$\Box\varphi \iff \bigwedge_{m=0}^{\infty} \circ^m\varphi \ .$$

By using previous results and explanations, one has "for free":

$$\xi(\Box\varphi) \triangleq \bigotimes_{m=0}^{\infty} \Big[\xi(\varphi) \rhd m\Big] \ . \tag{3.2}$$

This is exactly the approach which we shall follow, but there remains work to do in order to assign a precise meaning to (3.2), prove its correctness with respect to (3.1), and derive a closed form of (3.2) in terms of $\xi(\varphi)$, so that we may actually use it in an algorithm. In order to do so, we must first establish the legitimacy of using a *finite* extended notation

$$\bigotimes_{k=l}^{m} \sigma_k \triangleq \sigma_l \otimes \sigma_{l+1} \otimes \cdots \otimes \sigma_m$$

with the usual properties, which is provided by Remark 11.

Remark 11 (Extended Product Notation). The set of signatures Σ, equipped with the signature-product $\otimes$, forms a commutative monoid whose neutral element is ε.

Proof. The associativity and commutativity of $\otimes$ stem directly from those of $\cup$ and $\cap$. The neutrality of $\varepsilon = \wr \S \mathcal{R} \mid \overline{\mathbb{N}} \int$ stems from that of $\overline{\mathbb{N}}$ $(= \nabla\varepsilon)$ for $\cap$ within $\wp(\overline{\mathbb{N}})$, of $\mathcal{R}$ $(= \varepsilon[k]\,, \forall k)$ for $\cap$ within $\wp(\mathcal{R})$, and of $\varnothing$ (its domain) for $\cup$. □

All our previous results involving product behave as one would expect them to under extended notations; in particular, Lemma 7[p9] instantly generalises to

$$(\!|\Pi \S \bigotimes_{k=1}^{n} \sigma_k|\!) = \bigcap_{k=1}^{n} (\!|\Pi \S \sigma_k|\!) \qquad \forall \sigma_1, \ldots, \sigma_n \in \Sigma,\ n \in \mathbb{N}\,, \tag{3·3}$$

and it follows that Lemma 8[p9] becomes

$$\xi\left(\bigwedge_{k=1}^{n} \varphi_k\right) = \bigotimes_{k=1}^{n} \xi(\varphi_k) \qquad \forall \varphi_1, \ldots, \varphi_n \in \mathcal{A}\text{-LTL},\ n \in \mathbb{N}\,. \tag{3·4}$$

Infinite Products. This is still not enough, however, as (3.2) involves an *infinite* product, which is customarily defined based on finite products as follows: the infinite product $\bigotimes_{k=l}^{\infty} \sigma_k$ converges if and only if the associated sequence of partial products $\left(\bigotimes_{k=l}^{n} \sigma_k\right)_{n \in \mathbb{N}_l}$ converges, and in that case

$$\bigotimes_{k=l}^{\infty} \sigma_k \triangleq \lim_{n\to\infty} \bigotimes_{k=l}^{n} \sigma_k\,.$$

This definition rests upon the introduction of suitable notions of *convergence* and *limit* for sequences of signatures, to which we must now attend. Let $\rho = (\sigma_n)_{n\in\mathbb{N}}$ be an infinite sequence of signatures. It is said to be *convergent* if

(1) the sequence $(\nabla\sigma_n)_{n\in\mathbb{N}}$ converges towards a limit $\nabla\sigma_\infty$,
(2) for all $k \in \mathbb{N}_1$, the sequence $(\sigma_n[k])_{n\in\mathbb{N}}$ converges towards a limit $\sigma_\infty[k]$,
(3) the sequence of limits $(\sigma_\infty[k])_{k\in\mathbb{N}_1}$ itself converges towards a limit $\sigma_\infty[\infty]$.

We call the sequence $(\sigma_\infty[k])_{k\in\mathbb{N}_1}$ the *limit core*. It is not directly in the form of a bona fide signature core. However, its co-domain being $\wp(\mathcal{R})$, which is finite, there exists a rank $N \geqslant 0$ such that for all $k > N$, $\sigma_\infty[k] = \sigma_\infty[\infty]$, and thus, taking the smallest such N, we define the *limit* of ρ, which we denote by $\lim \rho$ or $\lim_{n\to\infty} \sigma_n$, or even more simply by σ_∞, as

$$\lim_{n\to\infty} \sigma_n \triangleq \wr \sigma_\infty[1]\,, \ldots, \sigma_\infty[N] \S \sigma_\infty[\infty] \mid \nabla\sigma_\infty \int\,.$$

Note that the core of the limit is equivalent [(f)] to the limit core, in the intuitive sense that they define the same constrained words. Otherwise ρ is *divergent*, and its limit is left undefined.

Example: Taking $\mathcal{R}_i = \mathcal{R}\ \forall i$, we have $\lim_{n\to\infty} \wr \mathcal{R}_1, \ldots \mathcal{R}_n, X \S \mathcal{R} \mid [\![1, n]\!] \int = \wr \S X \mid \mathbb{N}_1 \int$. It is easy to build artificial sequences that fail (1), (2) or (3). ♢

Without further ado, we can bring this notion of convergence to bear and further generalise (3.3) to infinitary cases. This result will be central to the proof of correctness.

Lemma 12 (*Breaking Infinite Products*)**.** *For any language* Π *and any sequence* $(\sigma_n)_{n\in\mathbb{N}}$ *of signatures such that the infinite product* $\bigotimes_{n=0}^{\infty} \sigma_n$ *converges,*

$$(\!|\Pi \S \bigotimes_{n=0}^{\infty} \sigma_n|\!) = \bigcap_{n=0}^{\infty} (\!|\Pi \S \sigma_n|\!)\,.$$

(f) The meticulous reader will have noticed that, throughout this section, we have omitted to mention that signatures are not unique, in the sense that for instance $\wr X \S \mathcal{R} \mid \overline{\mathbb{N}}_1 \int$, $\wr X, \mathcal{R} \S \mathcal{R} \mid \overline{\mathbb{N}}_1 \int$ and $\wr X, \mathcal{R}, \mathcal{R} \S \mathcal{R} \mid \overline{\mathbb{N}}_1 \int$ etcetera are all equally valid choices for $\xi(X)$. This slight ambiguity can easily be resolved by defining a notion of extensional equivalence between signatures and working with representatives of the classes. We have chosen to eschew this discussion entirely in the main text, as it is mostly a cosmetic consideration and would make the discussion more cumbersome. For future reference, extensional equivalence can be defined as $\sigma \equiv \rho \iff \nabla\sigma = \nabla\rho \wedge \forall k \in \mathbb{N}_1,\ \sigma[k] = \rho[k]$, and has the following fundamental characterisation: $\sigma \equiv \rho \iff \forall \Pi \subseteq \mathcal{T}, \forall \mathcal{R},\ (\!|\Pi \S \sigma|\!) = (\!|\Pi \S \rho|\!)$.

Proof. Let $\rho_m = \bigotimes_{n=0}^{m} \sigma_n$, for $m \in \overline{\mathbb{N}}$, and $\rho_\infty = \lim_{m\to\infty} \rho_m$. Let us compute the support of this limit

$$\nabla\rho_\infty = \lim_{m\to\infty} \nabla\rho_m = \lim_{m\to\infty} \nabla\left(\bigotimes_{n=0}^{m} \sigma_n\right) = \lim_{m\to\infty} \bigcap_{n=0}^{m} \nabla\sigma_n = \bigcap_{n=0}^{\infty} \nabla\sigma_n \ ,$$

and its limit core; let $k \in \mathbb{N}_1$ in

$$\rho_\infty[k] = \lim_{m\to\infty} \rho_m[k] = \lim_{m\to\infty} \left[\left(\bigotimes_{n=0}^{m} \sigma_n\right)[k]\right] = \lim_{m\to\infty} \bigcap_{n=0}^{m} \sigma_n[k] = \bigcap_{n=0}^{\infty} \sigma_n[k] \ .$$

So, a word $w \in (\!|\Pi \fatsemi \rho_\infty|\!)$ is a word of $(\!|\Pi|\!)$ such that

$$\begin{aligned}
&\#w \in \nabla\rho_\infty \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \rho_\infty[k] \\
\iff &\#w \in \textstyle\bigcap_{n=0}^{\infty} \nabla\sigma_n \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \bigcap_{n=0}^{\infty} \sigma_n[k] \\
\iff &\textstyle\bigwedge_{n=0}^{\infty} \left[\#w \in \nabla\sigma_n \ \wedge\ \forall k \in \operatorname{dom} w,\ w(k) \in \sigma_n[k]\right] \\
\iff &\textstyle\bigwedge_{n=0}^{\infty} w \in (\!|\Pi \fatsemi \sigma_n|\!) \iff w \in \bigcap_{n=0}^{\infty} (\!|\Pi \fatsemi \sigma_n|\!) \ .
\end{aligned}$$

□

We have just established that *provided that* the infinite product is convergent, we can break it, which will be as essential to the correctness proof as finite product breaking (Lem. 7) was to the proof of Lem. 8. There remains to prove that the computation of $\xi(\Box\,\varphi)$ always terminates and yields a useable signature.

Lemma 13 (*Shift-Product Convergence*)**.** *Let $(\sigma_n)_{n\in\mathbb{N}}$ be any sequence of signatures, and $(\rho_n)_{n\in\mathbb{N}}$ its associated sequence of partial products $(\bigotimes_{i=0}^{n} \sigma_i)_{n\in\mathbb{N}}$. Then $(\rho_n)_{n\in\mathbb{N}}$ satisfies convergence criteria* (1) *and* (2)*. Furthermore, if σ is a given signature and $\sigma_i = \sigma \blacktriangleright i$ or $\sigma_i = \sigma \rhd i$, for any $i \in \mathbb{N}$, then criterion* (3) *is satisfied as well, and the infinite product $\bigotimes_{n=0}^{\infty} \sigma_n$ converges.*

A detailed proof can be found in [6].

We now have every tool we need to write down the computation of $\xi(\Box\,\varphi)$.

Lemma 14 (*Always*)**.** *Provided that the signature of the subformula $\xi(\varphi)$ satisfies* (3.1)*, taking $\xi(\Box\,\varphi) \triangleq \bigotimes_{m=0}^{\infty} \left[\xi(\varphi) \rhd m\right]$ satisfies* (3.1)*.*

Proof. By application of Lem. 13, 12, $9_{[\text{p10}]}$ and $1_{[\text{p4}]}$. □

Example: Let us compute the signatures of two common $\Box$-based formulæ; although we could use the closed form given in the proof of Lemma 13 directly, we shall show a few manual steps in order to better see how the result is built:

$$\begin{aligned}
\xi(\Box\,X) &= \textstyle\bigotimes_{m=0}^{\infty} \left[\xi(X) \rhd m\right] = \bigotimes_{m=0}^{\infty} \left[\wr X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}}_1 \int \rhd m\right] \\
&= \textstyle\bigotimes_{m=0}^{\infty} \left[\wr \mathcal{R}_1, \ldots, \mathcal{R}_m, X \fatsemi \mathcal{R} \mid [\![0, m]\!] \cup (\overline{\mathbb{N}}_1 + m) \int\right] \\
&= \textstyle\bigotimes_{m=0}^{\infty} \left[\wr \mathcal{R}_1, \ldots, \mathcal{R}_m, X \fatsemi \mathcal{R} \mid [\![0, m]\!] \cup \overline{\mathbb{N}}_{1+m} \int\right] \\
&= \textstyle\bigotimes_{m=0}^{\infty} \left[\wr \mathcal{R}_1, \ldots, \mathcal{R}_m, X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}} \int\right] \\
&= \wr X \fatsemi X \mid \overline{\mathbb{N}} \int \equiv \wr \fatsemi X \mid \overline{\mathbb{N}} \int \ .
\end{aligned}$$

Note that the closed form by itself yields $\wr X \fatsemi X \mid \overline{\mathbb{N}} \int$, which could in this case be manually simplified into the "prettier" $\wr \fatsemi X \mid \overline{\mathbb{N}} \int$. These two signatures are obviously equivalent in the sense discussed in the footnote page 11, and we shall henceforth carry out similar simplifications as matter of course. As for the result itself, it is what was expected, as words of any length can satisfy $\Box\,\varphi$.

$$\begin{aligned}
\xi(\Box \bullet^1 X) &= \textstyle\bigotimes_{m=0}^{\infty} \left[\xi(\bullet^1 X) \rhd m\right] = \bigotimes_{m=0}^{\infty} \left[\wr \mathcal{R}, X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}}_2 \int \rhd m\right] \\
&= \textstyle\bigotimes_{m=0}^{\infty} \left[\wr \mathcal{R}_1, \ldots, \mathcal{R}_m, \mathcal{R}, X \fatsemi \mathcal{R} \mid [\![0, m]\!] \cup \overline{\mathbb{N}}_{2+m} \int\right] \\
&= \wr \mathcal{R} \fatsemi X \mid \{0\} \cup \{+\infty\} \int \ .
\end{aligned}$$

There again, the result should come as no surprise: the empty word satisfies $\Box \bullet^1 X$ vacuously, and as soon as there is one transition, then there must be another, and another… hence a word satisfying $\Box \bullet^1 X$ can only be either empty or infinite. ♢

$$\xi(\top) \triangleq \wr \S \mathcal{R} \mid \overline{\mathbb{N}} \S = \varepsilon \qquad \xi(\bot) \triangleq \wr \S \varnothing \mid \varnothing \S$$
$$\xi(X) \triangleq \wr X \S \mathcal{R} \mid \overline{\mathbb{N}}_1 \S \qquad \xi(\neg X) \triangleq \wr \mathcal{R} \setminus X \S \mathcal{R} \mid \overline{\mathbb{N}} \S$$
$$\xi(\bullet^m \varphi) \triangleq \xi(\varphi) \blacktriangleright m \qquad \xi(\circ^m \varphi) \triangleq \xi(\varphi) \rhd m$$
$$\xi(\varphi \wedge \psi) \triangleq \xi(\varphi) \otimes \xi(\psi) \qquad \xi(\Box \varphi) \triangleq \bigotimes_{m=0}^{\infty} \Big[\xi(\varphi) \rhd m \Big]$$

Fig. 2: Building Signatures on $\mathcal{A}$-LTL

Figure 2 and Thm. 15 summarise the eight main lemmata of this section.

Theorem 15 (*Signatures*). *For any $\Pi \subseteq \mathcal{T}$ and any $\varphi \in \mathcal{A}$-LTL, and given the map $\xi(\cdot) : \mathcal{A}\text{-LTL} \to \Sigma$ inductively defined in Fig. 2, it holds that*

$$(\!|\Pi \S \xi(\varphi)|\!) = \{\, w \in (\!|\Pi|\!) \mid w \models \varphi \,\} \ .$$

4. From Temporal Properties to Rewrite Propositions

The technical trek being now over, we come back to our overarching objective for the first problem, which is to translate temporal properties into rewrite propositions, and prove the translation's correctness. This will be accomplished by means of *translation rules* that are used in a way similar to the rules of a classical deduction system. Those rules are made up of *translation blocks*. We define the set $\mathfrak{B}$ of translation blocks as

$$\mathfrak{B} \triangleq \{\ \langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \ \mid \ \Pi \subseteq \mathcal{T},\ \sigma \in \Sigma,\ \varphi \in \mathrm{LTL} \,\} \ ,$$

where each translation block actually encodes a statement according to the following semantics:

$$\langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \quad \triangleq \quad \forall w \in (\!|\Pi \S \sigma|\!),\ w \models \varphi \ . \tag{4.1}$$

An *exact translation rule* is a statement of the form

$$\updownarrow \frac{\langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \qquad P(\sigma, \varphi)}{\pi} \quad \text{or} \quad \langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \, P(\sigma, \varphi) \quad \leftrightarrow \quad \pi \ ,$$

where the precondition $P \in \Sigma \times \mathrm{LTL} \to \mathbb{B}$ is a predicate on signatures and formulæ[(g)], and π is a mixed rewrite proposition, generated by the following grammar:

$$\pi := \gamma \mid \gamma \wedge \gamma \mid \gamma \vee \gamma \qquad \gamma := \ell = \varnothing \mid X \subseteq X \mid \ell \subseteq \ell \mid \Upsilon \qquad X \in \wp(\mathcal{R})$$
$$\ell := \Pi \mid \mathcal{T} \mid X(\ell) \mid X^{-1}(\ell) \mid X^*(\ell) \qquad \Upsilon \in \mathfrak{B} \ .$$

This is the grammar for rewrite propositions given in Sec. 2.3[p5], with the added production $\gamma := \Upsilon$, where Υ is a translation block. An exact translation rule has the following semantics:

$$\updownarrow \frac{\langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \qquad P(\sigma, \varphi)}{\pi} \quad \triangleq \quad P(\sigma, \varphi) \implies \big(\langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \Leftrightarrow \pi \big) \ .$$

In one instance, we shall give an *under-approximated translation rule*, written and defined in a similar manner as exact rules:

$$\uparrow \frac{\langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \qquad P(\sigma, \varphi)}{\pi} \quad \triangleq \quad P(\sigma, \varphi) \implies \big(\pi \Rightarrow \langle \Pi \S \sigma \ \Vdash \ \varphi \rangle \big) \ .$$

While *over-approximated translation rules* could obviously be defined as well, we have no use for them in the context of this paper, and the focus is markedly on exact translations. In the following, the unqualified words "translation", "rule" etcetera will always refer to *exact* translations and rules. The modus operandi of the (exact) translation of a formula $\varphi \in \mathcal{R}$-LTL consists in starting with the initial translation block $\langle \Pi \S \varepsilon \ \Vdash \ \varphi \rangle$, and transforming it by successive application of valid exact translation rules until we have a pure rewrite

(g) that will be omitted entirely if it is a tautology – which is the case in most rules.

proposition, that is to say until there are no translation blocks left at all. The resulting tree of rules, with $\langle \Pi \wr \varepsilon \Vdash \varphi \rangle$ at the root, will be called a *derivation*. By definition of the translation rules, this means that the rewrite proposition on the leaves of the derivation is equivalent to the initial translation block, and by the next theorem, that block is itself equivalent to the statement that the system $\mathcal{R}$, given the initial language Π, satisfies the property φ. In other words, it is an exact translation in the sense given in our problem statement, Sec. 2.3[p5]. Complete examples of derivations are given in Sec. 6.1[p25].

Theorem 16 (*Translation Satisfaction*)**.** $\langle \Pi \wr \varepsilon \Vdash \varphi \rangle \iff \mathcal{R}, \Pi \models \varphi$.

Proof. Recall that $(\!|\Pi \wr \varepsilon|\!) = (\!|\Pi|\!)$ by Lemma 2[p8];

$$\begin{aligned} \langle \Pi \wr \varepsilon \Vdash \varphi \rangle &\iff \forall w \in (\!|\Pi \wr \varepsilon|\!),\ w \models \varphi \iff \forall w \in (\!|\Pi|\!),\ w \models \varphi \\ &\iff \mathcal{R}, \Pi \models \varphi\ . \end{aligned}$$

□

Without further ado, we can begin to state and prove a few of the simplest translation rules. All rules given hereafter are theorems. We start with the simplest possible rule that can be given.

$$\langle \Pi \wr \sigma \Vdash \top \rangle \quad \leftrightarrow \quad \top \tag{$\top$}$$

Proof. We have by definition $\langle \Pi \wr \sigma \Vdash \top \rangle \Leftrightarrow \forall w \in (\!|\Pi \wr \sigma|\!),\ w \models \top \Leftrightarrow \top$. □

Although simple, this rule proves useful later on. Dealing with $\bot$ is both more delicate and less useful, so we leave it for the end.

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash X \wedge Y \rangle}{\langle \Pi \wr \sigma \Vdash X \cap Y \rangle} \tag{$\wedge_X$}$$

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash X \vee Y \rangle}{\langle \Pi \wr \sigma \Vdash X \cup Y \rangle} \tag{$\vee_X$}$$

Proof. This is a simple application of the semantics of $\wedge$ and X:

$$\begin{aligned} &\forall w \in (\!|\Pi \wr \sigma|\!),\ w \models X \wedge Y \\ \iff &\forall w \in (\!|\Pi \wr \sigma|\!),\ w \neq \lambda \ \wedge\ w(1) \in X \ \wedge\ w(1) \in Y \\ \iff &\forall w \in (\!|\Pi \wr \sigma|\!),\ w \models X \cap Y\ . \end{aligned}$$

Therefore by combining the empty and non-empty cases we obtain

$$\forall w \in (\!|\Pi \wr \sigma|\!),\ w \models X \wedge Y \iff \forall w \in (\!|\Pi \wr \sigma|\!),\ w \models X \cap Y\ .$$

Rule $(\vee_X)$[p14] is proven in the exact same way. □

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash \varphi \wedge \psi \rangle}{\langle \Pi \wr \sigma \Vdash \varphi \rangle \ \wedge\ \langle \Pi \wr \sigma \Vdash \psi \rangle} \tag{$\wedge$}$$

Proof. Conjunction distributes straightforwardly over the universal quantifier. □

As pointed out in Sec. 3.1[p5], disjunction does not enjoy the same privileges, and all we can do *in general* is state a crude under-approximated translation rule.

$$\uparrow \frac{\langle \Pi \wr \sigma \Vdash \varphi \vee \psi \rangle}{\langle \Pi \wr \sigma \Vdash \varphi \rangle \vee \langle \Pi \wr \sigma \Vdash \psi \rangle} \tag{$\vee_\uparrow$}$$

Please note however that $(\vee_\uparrow)$ will not be used in this paper, it is given here for the sake of completeness. Furthermore, there are some cases in which disjunction *can* be translated exactly; one such case appears in rule $(\vee_X)$, and two more useful cases will be introduced by the next few rules.

Recall that disjunction did not need to be handled in signatures at all (cf. Sec. 3.3[p7]), as this was best left to a translation rule. Specifically, disjunction in antecedents is handled by the following rule:

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash [\varphi \vee \varphi'] \Rightarrow \psi \rangle}{\langle \Pi \wr \sigma \Vdash \varphi \Rightarrow \psi \rangle \ \wedge\ \langle \Pi \wr \sigma \Vdash \varphi' \Rightarrow \psi \rangle} \tag{$\vee^{\Rightarrow}_{\wedge}$}$$

Proof. The result rests on the tautology $\big([\varphi \vee \varphi'] \Rightarrow \psi\big) \Leftrightarrow (\varphi \Rightarrow \psi) \wedge (\varphi' \Rightarrow \psi)$. The detailed steps are omitted. □

Note that neither ($\vee^{\Rightarrow}_{\wedge}$) nor similar rules based on tautologies – ($\vee_X$), ($\vee^{\neg}_{\Rightarrow}$) – are strictly necessary for the translation. They are nevertheless well worth a mention because they point out both limits and common modi operandi of the translation process. The next rule, called the *rule of signature introduction*, is essential to any non-trivial derivation, and rests upon the definition of $\xi(\cdot)$ given in the previous section.

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \ \Vdash \ \varphi \Rightarrow \psi \rangle \qquad \varphi \in \mathcal{A}\text{-LTL}}{\langle \Pi \fatsemi \sigma \otimes \xi(\varphi) \ \Vdash \ \psi \rangle} \tag{$\Rightarrow_\Sigma$}$$

Proof. We use the main property $(3.1)_{[p8]}$ of $\xi(\cdot)$ (cf. Theorem $15_{[p13]}$), as well as the (reverse) finite product-breaking Lemma $7_{[p9]}$.

$$\begin{aligned}
&\forall w \in (\!|\Pi \fatsemi \sigma|\!),\ w \models (\varphi \Rightarrow \psi) \iff \forall w \in (\!|\Pi \fatsemi \sigma|\!),\ (w \models \varphi) \Rightarrow (w \models \psi) \\
\iff &\forall w \in (\!|\Pi \fatsemi \sigma|\!),\ w \in (\!|\Pi \fatsemi \xi(\varphi)|\!) \Rightarrow w \models \psi \iff \forall w \in (\!|\Pi \fatsemi \sigma|\!) \cap (\!|\Pi \fatsemi \xi(\varphi)|\!),\ w \models \psi \\
\iff &\forall w \in (\!|\Pi \fatsemi \sigma \otimes \xi(\varphi)|\!),\ w \models \psi\ .
\end{aligned}$$ □

The signature-introduction rule makes it possible to handle disjunction in some more cases, as the next rule will show. Its proof rests on a tautology.

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \ \Vdash \ \varphi \vee \psi \rangle \qquad \exists \overline{\varphi} \in \mathcal{A}\text{-LTL} : \overline{\varphi} \Leftrightarrow \neg\varphi}{\langle \Pi \fatsemi \sigma \ \Vdash \ \overline{\varphi} \Rightarrow \psi \rangle} \tag{$\vee^{\neg}_{\Rightarrow}$}$$

While rule ($\vee^{\neg}_{\Rightarrow}$) is technically trivial, it has the merit of clearly showing the importance of the form in which a temporal formula is given. The best form to use is any form that allows an exact translation —there may be several.

We have now run out of translation rules that we can state and prove easily using only previously established results and definitions. Let us see what weakness or strength of context mean as far as signatures are concerned, by considering the least invasive operators that introduce a new context: the weak- and strong-next operators of level zero, $\circ^0$ and $\bullet^0$. If σ is the signature of some formula φ, then we have by definition of the weak and strong right shifts and by Lem. $9_{[p10]}$ and 10:

$$\xi\big(\circ^0\varphi\big) = \lceil \partial\sigma \mid \{0\} \cup \nabla\sigma \rfloor \quad \text{and} \quad \xi\big(\bullet^0\varphi\big) = \lceil \partial\sigma \mid \nabla\sigma \setminus \{0\} \rfloor\ .$$

In other words, what the context really changes is whether or not $0 \in \nabla\sigma$. We shall refer to a signature σ as *weak* if $0 \in \nabla\sigma$, and *strong* if $0 \notin \nabla\sigma$. This terminology is consistent with the operators of the above equations. We shall now argue that the mode of translation should *mirror* the quality of the signature in the current translation block. That is to say, given a block $\langle \Pi \fatsemi \sigma \ \Vdash \ \varphi \rangle$, the translation of φ should be strong if σ is weak, and weak if σ is strong. Thus we shall transition from a strong translation mode (or "strong context") to a weak mode (or "weak context") by "strengthening" the current translation block's signature: if σ is a signature, then $\star\sigma \triangleq \lceil \partial\sigma \mid \nabla\sigma \setminus \{0\} \rfloor$ is its *strengthening*. Let us apply this reasoning on the atomic cases of our translation: the literals $\neg X$ and X. Translating $\neg X$ is exactly like translating $\mathcal{R} \setminus X$ in a weak context.

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \ \Vdash \ \neg X \rangle}{\langle \Pi \fatsemi \star\sigma \ \Vdash \ \mathcal{R} \setminus X \rangle} \tag{$\neg X$}$$

Proof. The proof rests on the equality between the sets $(\!|\Pi \fatsemi \star\sigma|\!)$ and $(\!|\Pi \fatsemi \sigma|\!)^{\#}_{>0}$, which is justified by the definition of $(\!|\Pi \fatsemi \sigma|\!)$. □

The case of the atom is actually more complicated, and we leave it for the very last. We now deal with the translation of the weak next operator. The semantics suggests that, if the set of $(1{+}m)$-suffixes can be expressed as a set of constrained maximal rewrite words $(\!|\Pi' \fatsemi \sigma'|\!)$, then we shall simply need to ensure that those suffixes satisfy φ. The initial language Π' is immediately determined: $(1+m)$-suffixes are obtained after m rewriting steps from Π, performed according to σ. This is a very common pattern, and deserves a compact notation.

Signature Iteration. Let $\Pi \subseteq \mathcal{T}$ a language, and $\sigma \in \Sigma$ a signature; then for $n \in \mathbb{N}$ we let $\Pi^n_\sigma \triangleq$

$\sigma[n]\left(\sigma[n-1]\left(\cdots\sigma[1]\left(\Pi\right)\cdots\right)\right)$ be the *n-iteration* of the signature σ. More formally, it is defined recursively such that $\Pi_\sigma^0 \triangleq \Pi$ and $\Pi_\sigma^{n+1} \triangleq \sigma[n+1]\left(\Pi_\sigma^n\right)$.

With this notation, we have the initial language $\Pi' = \Pi_\sigma^m$. As for the signature σ', it is intuitively obtained by an operation which is dual to the right shifts: on each step the leftmost constraints of σ are "consumed" into the language. Naturally, we call this operation the *left shift*.

Arithmetic Overloading. We overload the operator $-$ on the profile $\wp(\overline{\mathbb{N}}) \times \mathbb{N} \to \wp(\overline{\mathbb{N}})$ such that, for any $S \in \wp(\overline{\mathbb{N}})$ and $n \in \mathbb{N}$, we have

$$S - n \triangleq \{\, k - n \mid k \in S \,\} \cap \overline{\mathbb{N}}\,.$$

Shift Left. Let $\sigma \in \Sigma$, $m \in \mathbb{N}$, then we define the *m-left shift of* σ as

$$\sigma \lhd m \triangleq \wr \partial\sigma(m+1), \ldots, \partial\sigma(\#\sigma) \fatsemi \partial\sigma(\omega) \mid \nabla\sigma - m \wr\,.$$

Note that we have, in a fashion dual to right shifts, the property that for all $m \in \mathbb{N}$ and all $k \in \mathbb{N}_1$, $(\sigma \lhd m)[k] = \sigma[k+m]$. However, this time there is no need to define weak and strong versions; instead, the strengthening star will be used whenever needed. One could nevertheless write the strong left shift as $\sigma \blacktriangleleft m \triangleq \star(\sigma \lhd m)$, as in [5].

Example: Let $\sigma = \wr X, Y \fatsemi Z \mid \mathbb{N}_2 \wr$; then $\sigma \blacktriangleleft 1 = \sigma \lhd 1 = \wr Y \fatsemi Z \mid \mathbb{N}_1 \wr$. ♢

Our earlier intuition about $(1+m)$-suffixes can now be formalised into the next lemma; recall that by definition of suffixes, $w^{\#w+1} = w^{\#w+2} = w^{\#w+3} = \cdots = \lambda$. It is therefore necessary to exclude too-short words, otherwise the empty word would have to appear in $(\!|\Pi_\sigma^m \fatsemi \sigma \lhd m|\!)$ not only when a term of Π_σ^m cannot be rewritten, but also if any term of some Π_σ^n, $n < m$, could not be rewritten.

Lemma 17 (*Shifting Suffixes*). *Let σ be a signature and $\Pi \subseteq \mathcal{T}$ a language; then* $(\!|\Pi_\sigma^m \fatsemi \sigma \lhd m|\!) = \left\{ w^{m+1} \;\middle|\; w \in (\!|\Pi \fatsemi \sigma|\!)^{\#}_{\geqslant m} \right\}$.

A detailed proof can be found in [6].

As announced, the translation rule is a forthright corollary of this lemma:

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \;\Vdash\; \circ^m \varphi \rangle}{\langle \Pi_\sigma^m \fatsemi \star(\sigma \lhd m) \;\Vdash\; \varphi \rangle} \qquad (\circ^m)$$

Dealing with the strong next operator is not much more difficult, thanks to: $w \models \bullet^m \varphi \iff \#w > m \wedge w \models \circ^m \varphi$. The only novelty here is the condition $\#w > m$, which will be translated by excluding smaller lengths.

Lemma 18. *Let σ be a signature and $\Pi \subseteq \mathcal{T}$ a language; then for any $m \in \mathbb{N}$, it holds that $(\!|\Pi \fatsemi \sigma|\!)^{\#}_m = \varnothing$ iff $m \in \nabla\sigma \implies \Pi_\sigma^m \subseteq \mathcal{R}^{-1}(\mathcal{T})$.*

A detailed proof can be found in [6]. Then, by Lemma 18[p16], we have:

Corollary 19 (*Length Rejection*). *Let $S \in \wp(\overline{\mathbb{N}})$, $\sigma \in \Sigma$ and $\Pi \subseteq \mathcal{T}$; it holds that $(\!|\Pi \fatsemi \sigma|\!)^{\#}_S = \varnothing$ iff $\bigwedge_{m \in S \cap \nabla\sigma} \Pi_\sigma^m \subseteq \mathcal{R}^{-1}(\mathcal{T})$ iff $\left(\bigcup_{m \in S \cap \nabla\sigma} \Pi_\sigma^m\right) \subseteq \mathcal{R}^{-1}(\mathcal{T})$.*

Given those results, the following translation can be proven:

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \;\Vdash\; \bullet^m \varphi \rangle}{\langle \Pi \fatsemi \sigma \;\Vdash\; \circ^m \varphi \rangle \;\wedge\; \bigwedge_{k \in [\![0,m]\!] \cap \nabla\sigma} \Pi_\sigma^k \subseteq \mathcal{R}^{-1}(\mathcal{T})} \qquad (\bullet^m)$$

The penultimate rules concern the $\Box$ operator; the approach is quite similar to that of Sec. 3.3. We start by writing

$$\langle \Pi \fatsemi \sigma \;\Vdash\; \Box\varphi \rangle \Leftrightarrow \left\langle \Pi \fatsemi \sigma \;\Vdash\; \bigwedge_{m=0}^{\infty} \circ^m \varphi \right\rangle \Leftrightarrow \bigwedge_{m=0}^{\infty} \langle \Pi \fatsemi \sigma \;\Vdash\; \circ^m \varphi \rangle\,,$$

and, after application of rule ($\circ^m$), we have

$$\langle \Pi \fatsemi \sigma \ \Vdash\ \Box\varphi \rangle \Leftrightarrow \bigwedge_{m=0}^{\infty} \left\langle \Pi_\sigma^m \fatsemi \star(\sigma \lhd m) \ \Vdash\ \varphi \right\rangle \ ,$$

but this falls short of a usable translation, at least in its current, infinite form. The usual intuition to overcome this kind of infiniteness is to hope for some kind of fixed point to be reached. Unfortunately, this is not the case here, since there is no reason in general to expect some h such that $\Pi_\sigma^h = \Pi_\sigma^{h+1}$. On the other hand, it seems reasonable that the signature may stabilise at some point, that is to say we might find some h such that $\star(\sigma \lhd h) = \star(\sigma \lhd (h+1))$, or more simply, h' such that $\sigma \lhd h' = \sigma \lhd (h'+1)$.

Stability. A signature $\sigma \in \Sigma$ is called *left-stable*, or simply *stable*, if it satisfies the following three, equivalent conditions:

(1) $\sigma \lhd 1 = \sigma$,
(2) $\forall n \in \mathbb{N},\ \sigma \lhd n = \sigma$,
(3) $\#\sigma = 0 \ \wedge\ \nabla\sigma \in \{\varnothing, \{+\infty\}, \mathbb{N}, \overline{\mathbb{N}}\}$.

By definition of the left shift, it is immediate that $(\sigma \lhd n) \lhd 1 = \sigma \lhd (n+1)$, and the first equivalence (1) $\Leftrightarrow$ (2) follows. The implication (3) $\Rightarrow$ (1) is also a simple application of the definition. To obtain the converse (1) $\Rightarrow$ (3), notice that if $\#\sigma > 0$ then $\#\sigma \lhd 1 = \#\sigma - 1 \neq \#\sigma$, but if $\#\sigma = 0$ then $\#\sigma \lhd 1 = \#\sigma = 0$. Similarly, we need to have

$$\begin{aligned} \nabla\sigma - 1 = \nabla\sigma &\iff \forall n \in \overline{\mathbb{N}}, \quad n \in \nabla\sigma - 1 \Leftrightarrow n \in \nabla\sigma \\ &\iff \forall n \in \overline{\mathbb{N}}, \quad n+1 \in \nabla\sigma \Leftrightarrow n \in \nabla\sigma \ . \end{aligned}$$

Since $\infty + 1 = \infty$, we can have either $\infty \in \nabla\sigma$ or $\infty \notin \nabla\sigma$; furthermore, if $0 \in \nabla\sigma$ then $\nabla\sigma \cap \mathbb{N} = \mathbb{N}$, and if $0 \notin \nabla\sigma$ then $\nabla\sigma \cap \mathbb{N} = \varnothing$. All in all, there are only the four possibilities listed in (3).

A stable signature allows for an easy translation of $\Box\varphi$, which can be stated as the rule ($\Box_*$). This rule, once proven, will serve as a lemma for the proof of the more general ($\Box_\hbar$), which subsumes it in the translation system.

$$\updownarrow \frac{\langle \Pi \fatsemi \sigma \ \Vdash\ \Box\varphi \rangle \qquad \sigma \text{ is stable}}{\left\langle \sigma[\omega]^*(\Pi) \fatsemi \star\sigma \ \Vdash\ \varphi \right\rangle} \tag{$\Box_*$}$$

A small intermediary remark is required for a complete proof:

Remark 20 (Constrained Union)**.** Let $\sigma \in \Sigma$, $I \subseteq \mathbb{N}$, and for each $i \in I$, $\Pi_i \subseteq \mathcal{T}$. Then $\bigcup_{i\in I} (\!|\Pi_i \fatsemi \sigma|\!) = (\!|\bigcup_{i\in I} \Pi_i \fatsemi \sigma|\!)$.

Proof. From the definition we immediately have $\bigcup_{i\in I} (\!|\Pi_i|\!) = (\!|\bigcup_{i\in I} \Pi_i|\!)$. Likewise, by definition $(\!|\Pi \fatsemi \sigma|\!) = \{ w \in (\!|\Pi|\!) \mid P(w,\sigma) \}$, where $P(w,\sigma)$ is some predicate depending only on w and σ. We have $\bigcup_{i\in I} (\!|\Pi_i \fatsemi \sigma|\!) = \bigcup_{i\in I} \{ w \in (\!|\Pi_i|\!) \mid P(w,\sigma) \} = \left\{ w \in \bigcup_{n\in I} (\!|\Pi_i|\!) \;\middle|\; P(w,\sigma) \right\} = \left\{ w \in (\!|\bigcup_{n\in I} \Pi_i|\!) \;\middle|\; P(w,\sigma) \right\} = (\!|\bigcup_{n\in I} \Pi_i \fatsemi \sigma|\!)$. □

Proof of rule ($\Box_*$). Assume that σ is stable.

$$\begin{aligned} &\langle \Pi \fatsemi \sigma \ \Vdash\ \Box\varphi \rangle \\ \Leftrightarrow &\textstyle\bigwedge_{m=0}^{\infty} \left\langle \Pi_\sigma^m \fatsemi \star(\sigma \lhd m) \ \Vdash\ \varphi \right\rangle \\ \Leftrightarrow &\forall m \in \mathbb{N},\ \forall w \in (\!|\Pi_\sigma^m \fatsemi \star\sigma|\!);\ w \models \varphi \\ \Leftrightarrow &\forall w \in (\!|\textstyle\bigcup_{m=0}^{\infty} \Pi_\sigma^m \fatsemi \star\sigma|\!);\ w \models \varphi \\ \Leftrightarrow &\forall w \in (\!|\sigma[\omega]^*(\Pi) \fatsemi \star\sigma|\!);\ w \models \varphi \end{aligned} \qquad \begin{aligned} &\langle \Pi \fatsemi \sigma \ \Vdash\ \Box\varphi \rangle \\ \Leftrightarrow &\textstyle\bigwedge_{m=0}^{\infty} \left\langle \Pi_\sigma^m \fatsemi \star\sigma \ \Vdash\ \varphi \right\rangle \\ \Leftrightarrow &\forall w \in \textstyle\bigcup_{m=0}^{\infty} (\!|\Pi_\sigma^m \fatsemi \star\sigma|\!);\ w \models \varphi \\ \Leftrightarrow &\forall w \in (\!|\textstyle\bigcup_{m=0}^{\infty} \sigma[\omega]^m(\Pi) \fatsemi \star\sigma|\!);\ w \models \varphi \\ \Leftrightarrow &\left\langle \sigma[\omega]^*(\Pi) \fatsemi \star\sigma \ \Vdash\ \varphi \right\rangle \ . \end{aligned}$$ □

In the next paragraphs, we explore how to get a stable signature from an unstable one, and how to employ that to effect the translation of $\Box\varphi$ in the general case.

High Point. The *high point* of a signature $\sigma \in \Sigma$, denoted by $\hbar\sigma$, is is defined according to either of the following equivalent statements:

(1) $\hbar\sigma \triangleq \min\{\, h \in \mathbb{N} \mid \sigma \lhd h \text{ is stable} \,\}$,
(2) $\hbar\sigma \triangleq \min\{\, h \in \mathbb{N}_{\#\sigma} \mid \nabla\sigma \supseteq \mathbb{N}_h \text{ or } \nabla\sigma \cap \mathbb{N}_h = \varnothing \,\}$.

The equivalence between those two definitions stems from the third characterisation of stability, because the stability of $\sigma \lhd \hbar\sigma$ entails $\#(\sigma \lhd \hbar\sigma) = 0$, which implies $\hbar\sigma \geqslant \#\sigma$, and $\nabla(\sigma \lhd \hbar\sigma) = \nabla\sigma - \hbar\sigma \in \{\varnothing, \{+\infty\}, \mathbb{N}, \overline{\mathbb{N}}\}$, hence $\nabla\sigma \supseteq \mathbb{N}_{\hbar\sigma}$ or $\nabla\sigma \cap \mathbb{N}_{\hbar\sigma} = \varnothing$. Note that since $\sigma \lhd 0 = \sigma$, the high point gives a fourth characterisation of stability – which is the most convenient one in practice – as σ is stable if and only if $\hbar\sigma = 0$. There remains, however, that not all signatures have a high point; consider the counterexamples $\sigma_1 = \wr X \mid \{\, 2k \mid k \in \mathbb{N} \,\}\wr$ or $\sigma_2 = \wr X \mid \mathbb{P}\wr$, where $\mathbb{P}$ is the set of prime numbers. We take the convention that in those cases $\hbar\sigma \triangleq +\infty$, and say that a signature σ is *stabilisable* if $\hbar\sigma \in \mathbb{N}$. It is fortunate that, while all signatures of Σ may not be stabilisable, in practice this is the case for all the signatures we shall need to deal with, as the next lemma will show.

Lemma 21 (*Stability of $\xi(\cdot)$*)**.** *The signature of any formula $\varphi \in \mathcal{A}$-LTL is stabilisable; in other words, $\hbar\xi(\varphi) \in \mathbb{N},\ \forall\varphi \in \mathcal{A}$-LTL.*

A detailed proof can be found in [6].

With this in place, the solution to a general translation of $\Box\varphi$ is suggested by the second statement of Lem. $1_{[p4]}$. The conjunction of weak-next operators needs only be unwound up to a certain, arbitrary rank, and there always exists a rank beyond which the signature stabilises, i.e., beyond which translation is no longer a problem. Hence we have the following complete rule, which supersedes the less general rule $(\Box_*)_{[p17]}$:

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash \Box\varphi \rangle \qquad \hbar\sigma \in \mathbb{N}}{\left\langle \Pi \wr \sigma \Vdash \bigwedge_{k=0}^{\hbar\sigma-1} \circ^k\varphi \right\rangle \wedge \langle \sigma[\omega]^*(\Pi_\sigma^{\hbar\sigma}) \wr \star(\sigma \lhd \hbar\sigma) \Vdash \varphi \rangle} \qquad (\Box_\hbar)$$

Proof. A corollary of Lem. $1_{[p4]}$'s second statement and of rules $(\Box_*)_{[p17]}$, $(\wedge)_{[p14]}$, $(\top)_{[p14]}$ and $(\circ^m)_{[p16]}$. □

Note that rule $(\top)$ is used when $\hbar\sigma = 0$, in which case the conjunction $\bigwedge_{k=0}^{\hbar\sigma-1} \circ^k\varphi$ is simply $\top$. Unlike $\top$, $\bot$ is never introduced by the rules themselves, since we never have to deal with potentially empty disjunctions. Also, termination can be enforced through other means (the atom $\varnothing$, for instance). Nevertheless, for the sake of completeness, we give a sketch of what the translation rule would be like. In the following, the map $\xi^{-1}(\cdot) : \Sigma \to \mathcal{A}$-LTL acts as an inverse (up to equivalence) for our signature-builder $\xi(\cdot)$. More specifically, it satisfies the conditions $\xi\big(\xi^{-1}(\sigma)\big) \equiv \sigma$ and $w \models \xi^{-1}(\xi(\varphi)) \Leftrightarrow w \models \varphi$.

$$\updownarrow \frac{\langle \Pi \wr \sigma \Vdash \bot \rangle}{\langle \Pi \wr \varepsilon \Vdash \xi^{-1}(\sigma) \rangle} \qquad (\bot)$$

Proof. This rests on the first-order tautology $\forall_x, (P(x) \Rightarrow \bot) \Leftrightarrow \forall_x, \neg P(x)$. □

It should be noted that $\xi^{-1}(\sigma)$ does not necessarily yield a translatable formula, so using $(\bot)$ brings no advantage compared to preprocessing. Thus it remains best to remove $\bot$s before the translation.

The very last case of this section is that of the atom X, the trickiest to translate. Let us recall two previous results in the case $\sigma = \varepsilon$:

$\langle \Pi \wr \varepsilon \Vdash \neg X \rangle \Leftrightarrow \langle \Pi \wr \star\varepsilon \Vdash \mathcal{R} \setminus X \rangle$ — $(\neg X)_{[p15]}$, $\sigma = \varepsilon$

$\langle \Pi \wr \varepsilon \Vdash \neg X \rangle \Leftrightarrow X(\Pi) = \varnothing$ — 1, Sec. $3.1_{[p5]}$.

The substitution of $\mathcal{R} \setminus X$ for X in the right-hand sides of the above equivalences immediately yields the following translation of the atom X in the case when $\sigma = \star\varepsilon$:

$$\langle \Pi \wr \star\varepsilon \Vdash X \rangle \Leftrightarrow [R \setminus X](\Pi) = \varnothing\ . \qquad (X_\varepsilon^\star)$$

Bearing in mind the first two cases of Sec. $3.1_{[p5]}$, it is π_2. As seen then, an additional condition was needed to ensure existence of the transition: $\Pi \subseteq \mathcal{R}^{-1}(\mathcal{T})$. With the additional notions and notations introduced since then, we can couch that by:

$$\updownarrow \frac{\langle \Pi \wr \varepsilon \Vdash X \rangle}{\langle \Pi \wr \star\varepsilon \Vdash X \rangle \ \wedge\ \Pi \subseteq \mathcal{R}^{-1}(\mathcal{T})}\ . \qquad (X_\varepsilon)$$

A concrete way of interpreting the above rule is to say that it exchanges the presence of 0 in the support of the signature for a statement rejecting the existence of 0-length maximal rewrite words. As for the "weak" part of the translation, $[R \setminus X](\Pi) = \varnothing$, it excludes all words of length 1 or more that do not start with a rule of X. While this partition of lengths may appear artificial in this case, it becomes more clearly marked in the next example. Let us consider the formula

$$\varphi = X \wedge \bullet^1 Y \wedge \bullet^2 Z \implies A \,.$$

The intuition under its translation is to generate the assertion that any maximal rewrite word which satisfies the antecedent, but does not satisfy the consequent, cannot exist. What would such a word look like? Starting with the initial language Π, it is obtained by successive applications of X, Y, and Z, followed by arbitrary many other applications of any rule in $\mathcal{R}$. Furthermore, its first rule is not in A. In other words, any word built on $Z\,(Y\,([X \setminus A](\Pi)))$ would satisfy those criteria. Thus we have the following translation:

$$\pi \equiv Z\,(Y\,([X \setminus A](\Pi))) = \varnothing \,.$$

Notice that this excludes only the words of length 3 or more which are built according to the succession $X \setminus A$, Y, Z. It is perfectly possible to have, for instance a maximal word $w = \rho$ of length 1, with $\rho \in X \setminus A$ and $t_0 \in \Pi \xrightarrow{\rho} t_1$. While it may violate the consequent, w does not actually satisfy the antecedent: because of the strong next of level 2 $\bullet^2$, a length of at least 3 is required for that —in terms of support, we have $\overline{\mathbb{N}}_3$. Supposing now that we had to deal with $\varphi'' = X \wedge \bullet^1 Y \wedge \circ^2 Z \Rightarrow A$, then the translation would be:$\pi'' \equiv Z\,(Y\,([X \setminus A](\Pi))) = \varnothing \wedge Y\,([X \setminus A](\Pi)) \subseteq \mathcal{R}^{-1}(\mathcal{T})$. At this point, the general method has become clear: if, assuming $\neg A$ for the first move, a length is in the support, and the words so built could be extended into something that violates the antecedent, then reject it by enforcing rewritability. However, if no possible extension of the words could violate the antecedent, then assert emptiness of its target language. Recalling the discussion of stability made earlier, that is certainly the case if the signature stabilises onto ε. In order to write in a compact way the assumption that the first transition is not by A, we overload the set difference operator $\setminus$ on the profile $\Sigma \times \wp(\mathcal{R}) \to \Sigma$ such that, for any $\sigma \in \Sigma$ and $X \subseteq \mathcal{R}$, we have

$$\sigma \setminus X \triangleq \sigma \otimes \xi(\neg X) \equiv \wr \sigma[1] \setminus X, \sigma[2]\,, \sigma[3]\,, \ldots, \sigma[\min(\#\sigma, 1)] \S\, \partial\sigma(\omega) \mid \nabla\sigma \wr \,.$$

We can now write the translation rule in the case where stabilisation is done on ε:

$$\updownarrow \frac{\langle \Pi \S \sigma \Vdash X \rangle \qquad (\sigma \setminus X) \lhd \hbar\,(\sigma \setminus X) = \varepsilon}{\Pi_{\sigma\setminus X}^{\hbar(\sigma\setminus X)} = \varnothing \quad \wedge \bigwedge_{k \in \nabla\sigma, k=0}^{\hbar(\sigma\setminus X)-1} \Pi_{\sigma\setminus X}^{k} \subseteq \mathcal{R}^{-1}\,(\mathcal{T})} \tag{$X_\hbar$}$$

Note that rules ($X_\varepsilon^\star$) and (X_ε) are in fact special cases of the above. The proof of this formula will be done with the help of the following small lemma:

Lemma 22. *Let $\sigma \in \Sigma$ and $h \in \mathbb{N}$ such that $\sigma \lhd h = \varepsilon$; then $(\!|\Pi \S \sigma|\!)^{\#}_{\geqslant h} = \varnothing$ iff $\Pi^h_\sigma = \varnothing$.*

Proof. It suffices to characterise that property in terms of h-suffixes:

$$(\!|\Pi \S \sigma|\!)^{\#}_{\geqslant h} = \varnothing \iff \left\{ w^{h+1} \;\middle|\; w \in (\!|\Pi \S \sigma|\!)^{\#}_{\geqslant h} \right\} = \varnothing \,.$$

Therefore, by Lem. 17[p16], we obtain $(\!|\Pi \S \sigma|\!)^{\#}_{\geqslant h} = \varnothing \Leftrightarrow (\!|\Pi^h_\sigma \S \sigma \lhd h|\!) = \varnothing \Leftrightarrow (\!|\Pi^h_\sigma \S \varepsilon|\!) = \varnothing \Leftrightarrow (\!|\Pi^h_\sigma|\!) = \varnothing \Leftrightarrow \Pi^h_\sigma = \varnothing$. □

Proof of rule ($X_\hbar$). By application of Lem. 7[p9] one has:

$$\begin{aligned} &\forall w \in (\!|\Pi \S \sigma|\!),\ w \models X \iff \forall w \in (\!|\Pi \S \sigma|\!),\ w \not\models \neg X \\ \iff &\forall w \in (\!|\Pi \S \sigma|\!),\ w \notin (\!|\Pi \S \xi(\neg X)|\!) \iff (\!|\Pi \S \sigma|\!) \cap (\!|\Pi \S \xi(\neg X)|\!) = \varnothing \\ \iff &(\!|\Pi \S \sigma \otimes \xi(\neg X)|\!) = \varnothing \iff (\!|\Pi \S \sigma \setminus X|\!) = \varnothing \,. \end{aligned}$$

It then becomes possible to reason on the length of the words; most of the work is done by invoking Lem. 22[p19]

and Cor. 19[p16];

$$\begin{aligned}
&(\!|\Pi \wr \sigma \setminus X|\!) = \varnothing \iff \forall k \in \overline{\mathbb{N}},\ (\!|\Pi \wr \sigma \setminus X|\!)^{\#}_{k} = \varnothing \\
\iff &(\!|\Pi \wr \sigma \setminus X|\!)^{\#}_{\geqslant \hbar(\sigma\setminus X)} = \varnothing \quad \wedge \quad (\!|\Pi \wr \sigma \setminus X|\!)^{\#}_{<\hbar(\sigma\setminus X)} = \varnothing \\
\iff &\Pi^{\hbar(\sigma\setminus X)}_{\sigma\setminus X} = \varnothing \quad \wedge \quad \textstyle\bigwedge_{k\in\nabla\sigma, k=0}^{k=\hbar(\sigma\setminus X)-1} \Pi^{k}_{\sigma\setminus X} \subseteq \mathcal{R}^{-1}(\mathcal{T})\ .
\end{aligned}$$ □

Cases when the signature does not stabilise onto ε – which correspond to $\sigma(\omega) \subsetneq \mathcal{R}$ – require an approximated approach to the same extent as a translation of $\Diamond$ does, and thus exceeds the scope of the present paper. Complete examples of derivations using the rules of this section are given in Sec. 6.1[p25].

5. Generating a Positive Approximation Procedure

We now have the tools to translate the original verification problem into an equivalent rewrite proposition, for a large gamut of temporal properties. The difference between the original undecidable problem statement in terms of a system and a temporal property, and its reformulation as a rewrite proposition, is that the latter is much more amenable to being transformed into a positive approximation procedure. The reader will have noticed that, in the previous section, the focus rested entirely on the temporal property φ, while decidability and representation of the languages involved by automata were ignored. In this section, we focus solely on those aspects.

5.1. A Simple Functional Algorithm

Our objective here is to translate a rewrite proposition π into a decision or positive approximation procedure δ – we call this operation "procedure generation". For the sake of clarity, we attribute a truth value to δ, which is computable and conflated to the result of its execution, that is to say, such that δ is true if the execution of the procedure generates a positive answer, and false if it does not answer. With this convention, we have $\delta \Rightarrow \pi$; this fits into our overall objective. Since π is *at worst* an under-approximated translation, we have in any case $\pi \implies \mathcal{R}, \Pi \models \varphi$; thus a positive answer of δ is enough to conclude that the system satisfies the expected property. More precisely, what will really be generated is not merely a single procedure δ, but a set of different theorems of the form "under such assumptions on the rewrite system, δ decides (resp. positively approximates) π".

The grammar of positive approximation procedures is quite similar to that of rewrite propositions in Sec. 2.3[p5]. Their semantics, although linked, are very different.

$$\begin{aligned}
\delta &:= \gamma \mid \gamma \wedge \gamma \mid \gamma \vee \gamma & \gamma &:= \alpha = \varnothing \mid \alpha \subseteq \alpha & X &\in \wp(\mathcal{R})\ . \\
& & \alpha &:= \Pi \mid \mathcal{T} \mid X(\alpha) \mid X^{-1}(\alpha) \mid X^{\star}(\alpha) \mid \natural\alpha
\end{aligned}$$

In the context of a positive approximation procedure, the notation $\Pi \mid \mathcal{T} \mid X(\ell) \mid X^{-1}(\ell)$ stands for an arbitrary tree automaton that accepts this same language. Two *kinds* of automata are considered in this paper: vanilla tree automata (TA) and tree automata with global equality constraints ($\mathsf{TA}^{=}$), but the method can certainly be extended to involve other varieties. Consequently, in the context of positive approximation procedures the cases $\ell = \varnothing \mid \ell \subseteq \ell$ designate either decision or positive approximation procedures for emptiness and inclusion (respectively) of the automata involved. Note that this overloading of notation does not introduce any ambiguity, so long as the context is kept clear. This brings us to the cases $X^{\star}(\alpha) \mid \natural\alpha$, which are unique to positive approximation procedures[(h)]. The crux of the undecidability of π is the potential presence of $X^{*}(\ell)$; however there are well-known methods [10, 8] which, given a regular language ℓ, compute a TA recognising a regular over-approximation (that is to say, a regular superset) of $X^{*}(\ell)$, which is what we denote by $X^{\star}(\alpha)$. As for $\natural\alpha$, which we call the "constraint relaxation of α", it is simply the TA obtained by removing all the equality constraints from the $\mathsf{TA}^{=}$ α. It is immediate that $\mathcal{L}(\natural\alpha) \supseteq \mathcal{L}(\alpha)$, thus we have a regular over-approximation again —but a very crude one, this time. Relaxations will be avoided inasmuch as possible.

There are implicit sanity rules which must be respected in order for the generated δ to be a valid positive approximation procedure. For instance, $\natural\alpha \subseteq \beta$ positively approximates $\mathcal{L}(\alpha) \subseteq \mathcal{L}(\beta)$, but $\alpha \subseteq \natural\beta$ does not.

(h) Note the different stars: $*$ in ℓ versus $\star$ in α.

Granted, that would be a silly way of positively approximating inclusion; it is merely an example to illustrate that one has to be careful to use over-approximations only where it does not break the positive approximation. Note that, as it depends in part upon the kinds of automata involved, it is best to deal with it separately, rather than attempting to incorporate those rules in the grammar of δ.

The question central to the positive approximation phase is the nature of an automaton α built to accept a language ℓ: could it simply be a TA? must it be a $\mathsf{TA}^=$? must it be something even more expressive? We can require of the user that the input Π be a regular tree language, and there is no question that $\mathcal{T}$ is regular in any case. It is also known that $X^{-1}(\mathcal{T})$ is accepted by a $\mathsf{TA}^=$ [4, Prp. 5], as well as $X(\ell)$, provided that ℓ is regular [4, Prp. 7]. Furthermore, properties of the rewrite systems can favourably influence the expressive power required: for instance, if X is left-linear, then $X^{-1}(\mathcal{T})$ is only a regular language. Moreover, there is a wide variety of properties of a rewrite system X under which $X^*(\ell)$ is actually regular, provided that ℓ itself is regular, see for instance [10]. Thus we can gain the following insights:

(1) The expressive power required to encode a language ℓ depends on a set of assumptions on the inductive sub-parts of ℓ. Each assumption belongs to one of three possible categories:

- **a.** the expressive power of a sub-language ℓ (regular, $\mathsf{TA}^=$, or beyond),
- **b.** linearity or other regularity-preserving properties of a TRS X,
- **c.** presence of an over-approximation.

(2) Procedure generation needs to build that set of assumptions inductively.

(3) Actually, in theory different combinations of assumptions may be chosen, which lead to potentially different procedures.

To illustrate this, let us consider $\ell = X(Y(\Pi))$. The sub-language Π is regular by hypothesis; $Y(\Pi)$ requires a $\mathsf{TA}^=$ (in general), thus ℓ cannot be expressed with a $\mathsf{TA}^=$. There are two possible paths: we can either make an assumption, or introduce an approximation. For instance, if we assume that Y preserves regularity through one-step rewriting (e.g. if it is linear), then $Y(\Pi)$ is regular, and it follows that ℓ is accepted by a $\mathsf{TA}^=$. Contrariwise, if we make no such assumption, we can still proceed by computing $\alpha^+ = X(\natural Y(\Pi))$, which is also a $\mathsf{TA}^=$, but then we have $\mathcal{L}(\alpha^+) \supseteq \ell$, and we need to keep in mind (i.e. add to our assumptions) that we are using an over-approximation. When considering the procedure generation in an abstract way, both those paths must be considered. Generally, the path which minimises the use of approximations and positive approximations will be considered the most desirable.

We shall deal with the question of expressive power and the related problem of detecting approximations by means of a simple inference system, referred to as "kind inference", working recursively on an automaton α built by generation from a rewrite language ℓ. Let $\mathfrak{K} \triangleq \{\, \mathsf{TA}, \mathsf{TA}^= \,\}$ be the set of all "kinds" of tree automata considered in this paper. Now let $\mathfrak{P} \triangleq \{\, \mathsf{left\text{-}lin}, \mathsf{reg\text{-}pres}, \mathsf{reg\text{-}pres}^* \,\}$ be the set of possible properties of a rewrite system, where $\mathsf{left\text{-}lin}$ stands for left-linear (regularity-preserving for backwards-rewriting of $\mathcal{T}$), $\mathsf{reg\text{-}pres}$ is a place-holder for any property or conjunction of properties that entail preservation of regularity through one-step forward rewriting (i.e. linear, etc), and $\mathsf{reg\text{-}pres}^*$ is a similar place-holder for preservation of regularity wrt. reachability (i.e. ground, right-linear & monadic, left-linear & semi-monadic, decreasing, etc [8]) [(i)]. Then we let $\mathfrak{A}$ be the set of all possible assumptions, defined as

$$\mathfrak{A} \triangleq \{\, \alpha : k \mid k \in \mathfrak{K} \,\} \uplus \{\, X : p \mid X \in \wp(\mathcal{R}), p \in \mathfrak{P} \,\} \uplus \{\, \alpha : + \,\} \uplus \{\, \gamma : + \,\} \ ,$$

where α is any automaton-expression and γ any test as defined by the grammar at the beginning of the section. The four disjoint sets in this definition correspond to the four kinds of assumptions listed earlier. In the case of a test γ, $\gamma : +$ means that it is a positive approximation instead of an exact test. A kind-inference rule is either *simple* or a *chain* of simple rules; a simple rule is of the form

$$\Gamma \vdash \Delta \qquad \text{where} \quad \Gamma, \Delta \in \wp(\mathfrak{A}) \ ,$$

and the meaning that, whenever all the assumptions within Γ hold, then so do all those within Δ. Regarding notations, the sets' braces will be omitted when writing the rules, and the comma is taken to mean set union:

[(i)] The practical implementation detects the specific properties that ensures this, which we abstract in this paper. Note that this aspect will keep improving as new regularity-preserving classes of TRS are discovered and implemented.

e.g. "Γ, Γ', a" means "$\Gamma \cup \Gamma' \cup \{a\}$". A chain-rule is a $\lhd$-separated sequence of simple rules,

$$\Gamma_1 \vdash \Delta_1 \lhd \Gamma_2 \vdash \Delta_2 \lhd \ldots \lhd \Gamma_n \vdash \Delta_n \ ,$$

such that $\Gamma_i \subseteq \Gamma_{i+1}$. A chain-rule behaves as one of the simple rules in the chain. Given the assumptions Γ, the rule that applies is $\Gamma_k \vdash \Delta_k$, such that $\Gamma_k \subseteq \Gamma$, and either $k = n$ or $\Gamma_{k+1} \not\subseteq \Gamma$. Chain-rules are useful in the common cases where making some more assumptions (on the left, e.g. about a rewrite system) prevents us from having to make some *other* assumptions (e.g. that such language is over-approximated). Given a set of rules S (simple and chains), and a set of assumptions Γ, *one-step deduction* is written $\Gamma \vdash^1 \Delta$, and holds iff there is in S either a simple rule $\Gamma' \vdash \Delta'$ such that $\Gamma' \subseteq \Gamma$ and $\Delta \subseteq \Delta'$, or a chain-rule whose active simple rule, given Γ, satisfies the same properties. A *deduction*, written $\Gamma_0 \vdash^* \Delta$, can be seen as extending this by reflexivity and transitivity, and is defined as follows:

$$\Gamma_0 \vdash^* \Delta \quad \text{iff} \quad \Delta \subseteq \Gamma_0 \ \vee \ \exists \Gamma_1, \ldots, \Gamma_n : \Gamma_n \supseteq \Delta \ \wedge \ \forall k \in [\![1, n]\!] \, , \ \textstyle\bigcup_{i=0}^{k-1} \Gamma_i \vdash^1 \Gamma_k \ .$$

Let us now state the axioms of the kind-inference system. We start by the most immediate ones:

$$\vdash \Pi : \mathsf{TA} \qquad\qquad \vdash \mathcal{T} : \mathsf{TA} \ .$$

The first rule is a practical hypothesis. The second is trivially true. A third basic rule one might want to state is $\alpha : \mathsf{TA} \vdash \alpha : \mathsf{TA}^=$, which would reflect the fact that TA are technically degenerate cases of $\mathsf{TA}^=$ where the set of constraints is empty. However, this rule is not included in the system, as the assumption $\alpha : \mathsf{TA}$ is taken to mean that α recognises a tree language that is known to be regular, while $\alpha : \mathsf{TA}^=$ means that α accepts a $\mathsf{TA}^=$-language that *may* be regular.

With this in mind, we move on to forward rewriting, for which there are two main cases in a chain-rule: either the rewrite system preserves regularity, or it does not, in which case we use a $\mathsf{TA}^=$ [4, Prp. 7]. Forward rewriting cannot a priori be done on a $\mathsf{TA}^=$-language while staying within the allowed kinds of tree automata (TA and $\mathsf{TA}^=$), therefore there is no rule in that case;

$$\alpha : \mathsf{TA} \vdash X(\alpha) : \mathsf{TA}^= \lhd \qquad\qquad \alpha : \mathsf{TA}, X : \mathsf{reg\text{-}pres} \vdash X(\alpha) : \mathsf{TA} \ .$$

Backwards rewriting is similarly captured by a two-rules chain, hinging on left linearity [4, Prp. 5]:

$$\vdash X^{-1}(\mathcal{T}) : \mathsf{TA}^= \lhd \qquad\qquad X : \mathsf{left\text{-}lin} \vdash X^{-1}(\mathcal{T}) : \mathsf{TA} \ .$$

Note that the more general case $X^{-1}(\ell)$ is not handled, because the derivations of Sec. 4[p13] are such that no translation rule can yield rewrite propositions that require it. Our first approximated case is constraint relaxation, with two independent rules:

$$\alpha : \mathsf{TA} \vdash \natural\alpha : \mathsf{TA} \qquad\qquad \alpha : \mathsf{TA}^= \vdash \natural\alpha : \mathsf{TA}, \natural\alpha : + \ .$$

By the first rule, relaxing the constraints of a TA gives a TA —the same as before. This rule could be omitted without damage. The second rule simply states that relaxing the constraints of a $\mathsf{TA}^=$ results in a TA, and introduces an over-approximation, in general [(j)]. Note that the two rules are not in competition, as $\alpha : \mathsf{TA}$ and $\alpha : \mathsf{TA}^=$ are contradictory assumptions. The other approximated case is reachability over-approximation, captured by a two-rules chain.

$$\alpha : \mathsf{TA} \vdash X^\star(\alpha) : \mathsf{TA}, X^\star(\alpha) : + \lhd \qquad\qquad \alpha : \mathsf{TA}, X : \mathsf{reg\text{-}pres}^* \vdash X^\star(\alpha) : \mathsf{TA} \ .$$

In the general case (first rule), the use of some over-approximation algorithm is required; however in the best case (second rule), it can safely be assumed that no approximation has taken place: $\mathcal{L}\left(X^\star(\alpha)\right) = X^*(\ell)$.

Lastly, we must not forget to encode the fact that over-approximation propagates through forward rewriting, and contaminates tests:

$$\alpha : + \vdash X(\alpha) : +, (\alpha = \varnothing) : +, (\alpha \subseteq \beta) : + \ ,$$

which in turn contaminate the entire procedure, confining it to positive approximation:

$$\gamma : + \vdash (\gamma \vee \gamma') : +, (\gamma \wedge \gamma') : + \ .$$

(j) One might construct special instances of $\mathsf{TA}^=$ where relaxation does not introduce any approximation, for instance if the constraints involve states that are unreachable.

Even without any prior approximation, an inclusion test must be a positive approximation procedure if its right-hand side is a $\mathsf{TA}^{=}$, and cannot be a TA; in that case the usual method ($\alpha \cap \beta^{c} = \varnothing$) cannot apply, since $\mathsf{TA}^{=}$ cannot be complemented [9].

$$\beta : \mathsf{TA}^{=} \vdash (\alpha \subseteq \beta) : + \ .$$

This system is not complete (and no implementation of it can be), but the rules above are quite sufficient for our immediate purposes. Indeed, we can now proceed to write the procedure generation itself. As mentioned before, the generation is to be done inductively on the structure of the input rewrite proposition. More precisely, given a rewrite proposition π, we want to obtain the set of all possible couples Δ, δ, such that $\Delta \subseteq \mathfrak{A}$ and δ is the best positive approximation procedure under the assumptions Δ. By "the best" we mean "minimising the use of approximations and positive approximations". Such a couple Δ, δ can be regarded as the theorem "If Δ, then δ positively approximates π.". We use a recursive *relation* given as a set of rules. Here is what a procedure-generation rule looks like in the most general case:

$$\Gamma \fatsemi [\ell_1 \rightarrowtail \Delta_1, \delta_1; \ell_2 \rightarrowtail \Delta_2, \delta_2; \dots] \fatsemi P \fatsemi \pi \rightrightarrows \Delta \fatsemi \delta \ .$$

Let us start by the parts that are not optional: $\Gamma \subseteq \mathfrak{A}$ is a set of assumptions, in practice only pertaining to properties of the rewrite system, i.e. excluding kind and over-approximation assumptions; π is the rewrite proposition (resp. sub-part of a rewrite proposition, such as a language ℓ or a comparison γ) being converted; δ is the corresponding procedure (resp. sub-part of a procedure, such as an automaton α or simple comparison γ), and Δ is the set of assumptions under which δ is constructed. The optional parameter P is a predicate which must be satisfied in order for the rule to apply; it is mostly used for kind inference statements. The optional list of patterns of the form "$\ell_k \rightarrowtail \Delta_k, \delta_k$" between brackets serves to name *possible* recursive calls; the ℓ_k are supposed to be direct sub-components of π, and Δ_k, δ_k corresponds to any one possible result of the procedure generation for ℓ_k, under the assumptions Γ. The simplest rules need neither of those options:

$$\Gamma \fatsemi \Pi \rightrightarrows \Gamma \fatsemi \Pi \qquad\qquad \Gamma \fatsemi \mathcal{T} \rightrightarrows \Gamma \fatsemi \mathcal{T} \ .$$

In both cases, the language on the left simply becomes the automaton on the right, and no assumption is required or introduced. The case of backwards rewriting is also quite simple, though it requires two rules:

$$\Gamma \fatsemi X^{-1}(\mathcal{T}) \rightrightarrows \Gamma \fatsemi X^{-1}(\mathcal{T}) \qquad\qquad \Gamma \fatsemi X^{-1}(\mathcal{T}) \rightrightarrows \Gamma, X : \mathsf{left\text{-}lin} \fatsemi X^{-1}(\mathcal{T}) \ .$$

In the first rule, no extra assumption is introduced, which means that the resulting automaton will need to be a $\mathsf{TA}^{=}$ in general. In the second rule, the introduction of the assumption $X : \mathsf{left\text{-}lin}$ means that it will only be a TA. Depending on which rule is chosen, the subsequent derivation will yield different procedures (the first choice may lead to constraints relaxations that are unneeded in the second, for instance), different assumptions, and thus different theorems. This is a very common pattern; in fact, that situation occurs at every point of the generation where the presence or absence of some properties of the rewrite system changes the required kind of the automaton. To avoid writing all the different possible rules manually, we "factor" them by putting such properties between angle-brackets: $\langle p_1, \dots, p_n \rangle$. A rule where this syntax appears is short for the 2^n rules obtained by choosing all possible subsets of those properties. Thus the last two rules can be written simply as:

$$\Gamma \fatsemi X^{-1}(\mathcal{T}) \rightrightarrows \Gamma, \langle X : \mathsf{left\text{-}lin} \rangle \fatsemi X^{-1}(\mathcal{T}) \ .$$

Forward rewriting needs to be more general than the backwards case, thus the next rules are recursive:

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^{*} \alpha : \mathsf{TA} \ \fatsemi X(\ell) \rightrightarrows \Gamma, \Delta, \langle X : \mathsf{reg\text{-}pres} \rangle \fatsemi X(\alpha)$$
$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^{*} \alpha : \mathsf{TA}^{=} \fatsemi X(\ell) \rightrightarrows \Gamma, \Delta, \langle X : \mathsf{reg\text{-}pres} \rangle \fatsemi X(\natural\alpha) \ .$$

The first rule means that, given the assumptions Γ, and further assuming that the language ℓ, under the same assumptions Γ, can be converted into the automaton α, with the resulting assumptions Δ, and also assuming that α can be a simple TA, the language $X(\ell)$ can be represented by the automaton $X(\alpha)$, under the union of our starting assumptions Γ, and the assumptions Δ made during the generation of α. Furthermore, whether or not X is regularity-preserving influences the kind of automaton obtained, and thus leads to different branches. Note that explicitly returning the union $\Gamma \cup \Delta$, as was done here, is not strictly necessary, since we take care that no rule ever removes any assumptions – they can only be added, – and thus $\Delta \supseteq \Gamma$.

The second rule deals with the case where the automaton α can only be a $\mathsf{TA}^{=}$; in that case, we need to make an over-approximation by relaxing the constraints of α before forward rewriting. Then again, different

branches must be explored depending on whether X is regularity-preserving. We have the same kind of pattern for reachability:

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^* \alpha : \mathsf{TA} \quad \fatsemi X^*(\ell) \rightrightarrows \Gamma, \Delta, \langle X : \textsf{reg-pres}^* \rangle \fatsemi X^\star(\alpha)$$
$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^* \alpha : \mathsf{TA}^= \fatsemi X^*(\ell) \rightrightarrows \Gamma, \Delta, \langle X : \textsf{reg-pres}^* \rangle \fatsemi X^\star(\natural\alpha) \ .$$

Depending on whether X is regularity-preserving for reachability, $X^\star(\alpha)$ (resp. $X^\star(\natural\alpha)$) will be exact or over-approximated (resp. over- and twice-over–approximated). We are now done with the construction of automata, and move on to the generation of tests γ, starting with inclusion.

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha; \ell' \rightarrowtail \Delta', \alpha'] \fatsemi \Delta' \not\vdash^* \alpha' : + \fatsemi \ell \subseteq \ell' \rightrightarrows \Gamma, \Delta, \Delta' \fatsemi \alpha \subseteq \alpha' \ .$$

Recall that if $\alpha' : \mathsf{TA}^=$, then $(\alpha \subseteq \alpha') : +$ will be deduced by the other rules regarding approximations. The only other kind of simple test is emptiness testing:

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^* \alpha : \mathsf{TA} \vee \Delta \vdash^* \alpha : \mathsf{TA}^= \fatsemi \ell = \varnothing \rightrightarrows \Gamma, \Delta \fatsemi \alpha = \varnothing \ .$$

This is the immediate case: an automaton α accepting ℓ is built, and regardless of whether α has constraints or not, an emptiness test is run on it; the algorithmic complexity changes (ExpTime vs linear time), but this is a decision procedure in both cases. There is another case, where it is possible to test emptiness of a language *without* being able to build the corresponding automaton [4, Prp. 6], given by the next rule.

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^* \alpha : \mathsf{TA}^= \fatsemi X(\ell) = \varnothing \rightrightarrows \Gamma, \Delta \fatsemi X(\alpha) = \varnothing \ .$$

In that case $X(\alpha)$ cannot be built, since it is not even known whether it is at all recognisable by a $\mathsf{TA}^=$. One possibility, covered by previous rules, is to actually test $X(\natural\alpha) = \varnothing$, which introduces an approximation; but in that case it is a terrible idea, as $X(\ell) = \varnothing$ iff $\ell \cap \mathcal{R}^{-1}(\mathcal{T}) = \varnothing$, which is decidable thanks to $\mathsf{TA}^=$ being closed by intersection. The above rule reflects that fact, and "$X(\alpha) = \varnothing$," where α is a $\mathsf{TA}^=$, is taken to abstract the above test. Note that no approximation will be deduced for $X(\alpha)$; in fact, the previous rules simply have nothing to say about $X(\alpha)$, since it is not an automaton of any allowed kind, but merely a notation that only takes meaning in the context of an emptiness test. Again, this could be reflected explicitly in the grammar, at the cost of introducing new symbols. Lastly, let us emphasise that both paths (exact intersection-emptiness test and constraint relaxation) will be explored by the system, and yield two different theorems. It is easy to discriminate between the two, as only the assumptions generated by the second theorem will enable the deduction of $(X(\alpha) = \varnothing) : +$, and not the first. In a practical implementation, the best option is always chosen if this special rule is applied with a higher priority than the more general forward-rewriting rules. Before concluding, let us not forget the rules for conjunctions and disjunctions of tests, which are trivial given our convention regarding the truth value of a positive approximation procedure, mentioned at the beginning of this section:

$$\Gamma \fatsemi [\gamma \rightarrowtail \Delta, \delta; \gamma' \rightarrowtail \Delta', \delta'] \fatsemi \gamma \wedge \gamma' \rightrightarrows \Gamma, \Delta, \Delta' \fatsemi \delta \wedge \delta'$$
$$\Gamma \fatsemi [\gamma \rightarrowtail \Delta, \delta; \gamma' \rightarrowtail \Delta', \delta'] \fatsemi \gamma \vee \gamma' \rightrightarrows \Gamma, \Delta, \Delta' \fatsemi \delta \vee \delta' \ .$$

6. Examples & Discussion of Applicability

The question of whether the proposed verification chain is applicable in the real world rests on two separate issues. The first, and most important, is whether the fragment of LTL which can be handled is actually sufficiently large to describe relevant properties of systems. The second is whether the quality of the resulting positive approximation procedures is likely to be acceptable. To address the first issue, we take a look at the kinds of temporal patterns which can be translated, and attempt to quantify how useful they might be, based on the comprehensive study done on a large number of specifications in [27]. The second issue is dependant on both the depth of the temporal formula that needs to be checked, and the properties of the rewrite system under consideration. Thus we look at some existing TRS models in the literature, specifically a model for the Java Virtual Machine and Java bytecode in [2], a model for the Needham–Schroeder protocol in [1], and a model for CCS specifications without renaming in [20]. As a prelude to these discussions, we give three examples of derivations, using the temporal patterns of [4].

6.1. Examples: Three Derivations

The three temporal properties below, while simple, are varied enough to test every main translation rule [(k)]. The first pattern, $\square(X \Rightarrow \bullet^1 Y)$, has already been illustrated in the introduction; its translation into a rewrite proposition is obtained through a straight-forward, five-step derivation:

$$\updownarrow \frac{\langle \Pi \fatsemi \varepsilon \Vdash \square(X \Rightarrow \bullet^1 Y)\rangle}{\updownarrow \dfrac{\langle \mathcal{R}^*(\Pi) \fatsemi \star\varepsilon \Vdash X \Rightarrow \bullet^1 Y\rangle}{\updownarrow \dfrac{\langle \mathcal{R}^*(\Pi) \fatsemi \wr X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}}_1 \int \Vdash \bullet^1 Y\rangle}{\updownarrow \dfrac{\langle \mathcal{R}^*(\Pi) \fatsemi \wr X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}}_1 \int \Vdash \circ^1 Y\rangle}{\updownarrow \dfrac{\langle X(\mathcal{R}^*(\Pi)) \fatsemi \star\varepsilon \Vdash Y\rangle}{[\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi))) = \varnothing}}}}} \begin{array}{r}(\square_*)\\(\Rightarrow_\Sigma)\\(\bullet^m)\\(\circ^m)\\(X^\star_\varepsilon)\end{array} \wedge X(\mathcal{R}^*(\Pi)) \subseteq \mathcal{R}^{-1}(\mathcal{T}) \ .$$

The result can then be optimised into $[\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi))) = \varnothing \ \wedge \ X(\mathcal{R}^*(\Pi)) \subseteq Y^{-1}(\mathcal{T})$, which is the expected final translation.

For the procedure generation, as there are quite a lot of choices, the resulting tree structure is hard to represent on paper. Instead, we will effect the generation "from the inside out", unwinding the recursivity and keeping track of the branching possibilities manually. Let us deal with $[\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi))) = \varnothing$ first; we build the automaton accepting the language, step by step, and without making any a priori assumptions: $\Gamma = \varnothing$. Our first step is the initial language: the only possibility is $\varnothing \fatsemi \Pi \rightrightarrows \varnothing \fatsemi \Pi$. The second step is $\mathcal{R}^*(\Pi)$, which is handled by the rule for reachability:

$$\varnothing \fatsemi [\Pi \rightarrowtail \varnothing, \Pi] \fatsemi \varnothing \vdash^* \Pi : \mathsf{TA} \fatsemi \mathcal{R}^*(\Pi) \rightrightarrows \langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle \fatsemi \mathcal{R}^\star(\Pi) \ .$$

Then either X is regularity-preserving for reachability or it is not. Third step: $X(\mathcal{R}^*(\Pi))$, using the rule for forward-rewriting. Let us explore the first branch, where $\langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle = \varnothing$:

$$\begin{aligned}\varnothing \fatsemi [\mathcal{R}^\star(\Pi) \rightarrowtail \varnothing, \mathcal{R}^\star(\Pi)] \fatsemi \varnothing \vdash^* \mathcal{R}^\star(\Pi) : \mathsf{TA} \fatsemi X(\mathcal{R}^\star(\Pi))\\ \rightrightarrows \langle X : \mathsf{reg\text{-}pres} \rangle \fatsemi X(\mathcal{R}^\star(\Pi)) \ .\end{aligned}$$

Now, the second branch, where $\langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle = \{\mathcal{R} : \mathsf{reg\text{-}pres}^*\}$, which we write Δ:

$$\begin{aligned}\varnothing \fatsemi [\mathcal{R}^\star(\Pi) \rightarrowtail \Delta, \mathcal{R}^\star(\Pi)] \fatsemi \Delta \vdash^* \mathcal{R}^\star(\Pi) : \mathsf{TA} \fatsemi X(\mathcal{R}^\star(\Pi))\\ \rightrightarrows \Delta, \langle X : \mathsf{reg\text{-}pres} \rangle \fatsemi X(\mathcal{R}^\star(\Pi)) \ .\end{aligned}$$

In both cases, this creates a new branch, depending on whether X is regularity-preserving for one step rewriting, which one could summarise as

$$\langle \mathcal{R} : \mathsf{reg\text{-}pres}^*, X : \mathsf{reg\text{-}pres} \rangle , X(\mathcal{R}^*(\Pi)) \ .$$

On to the fourth step; this time, there are two possible rules that may apply. One could go on and compute the automaton for $[\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi)))$, applying the forward-rewriting rule again; however, this is not possible in all branches. To simplify, let us state that $\mathcal{R} : \mathsf{reg\text{-}pres}^*$ is moot at this point. On the other hand, $X : \mathsf{reg\text{-}pres}$ influences the kind of automaton generated for $X(\mathcal{R}^*(\Pi))$, thus we really just need to discriminate between $\Delta \in \langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle$ and $\Delta \in \{X : \mathsf{reg\text{-}pres}\} \cup \langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle$ – writing Δ the assumptions accumulated in the current branch. In the first case(s), we have $\Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA}^=$, which requires the use of a constraint relaxation:

$$\begin{aligned}\varnothing \fatsemi [X(\mathcal{R}^*(\Pi)) \rightarrowtail \Delta, X(\mathcal{R}^\star(\Pi))] \fatsemi \Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA}^= \fatsemi\\ [\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi))) \rightrightarrows \varnothing, \Delta, \langle \mathcal{R} \setminus Y : \mathsf{reg\text{-}pres} \rangle \fatsemi [\mathcal{R} \setminus Y](\natural X(\mathcal{R}^\star(\Pi))) \ .\end{aligned}$$

In the second case(s), we have $\Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA}$, which obviates the need for such:

$$\begin{aligned}\varnothing \fatsemi [X(\mathcal{R}^*(\Pi)) \rightarrowtail \Delta, X(\mathcal{R}^\star(\Pi))] \fatsemi \Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA} \fatsemi\\ [\mathcal{R} \setminus Y](X(\mathcal{R}^*(\Pi))) \rightrightarrows \varnothing, \Delta, \langle \mathcal{R} \setminus Y : \mathsf{reg\text{-}pres} \rangle \fatsemi [\mathcal{R} \setminus Y](X(\mathcal{R}^\star(\Pi))) \ .\end{aligned}$$

Overall, this creates a third binary branching at this step; but there is more. There is another possibility: one could use the special rule for testing emptiness without actually attempting to compute the automaton.

(k) The unused rules consist of the tautology-based simplifications, such as $(\vee^{\Rightarrow}_{\wedge})_{[\text{p14}]}$, $(\vee^{\neg}_{\Rightarrow})_{[\text{p15}]}$, $(\wedge_X)_{[\text{p14}]}$, $(\vee_X)$, etcetera.

While this could in principle be done in every case, the rule only applies if $\Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA}^=$ and we have thus, for $\Delta \in \langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle$:

$$\varnothing \fatsemi [X(\mathcal{R}^*(\Pi)) \rightarrowtail \Delta, X(\mathcal{R}^\star(\Pi))] \fatsemi \Delta \vdash^* X(\mathcal{R}^\star(\Pi)) : \mathsf{TA}^= \fatsemi$$
$$[\mathcal{R} \setminus Y]\,(X(\mathcal{R}^*(\Pi))) = \varnothing \rightrightarrows \Delta \fatsemi [\mathcal{R} \setminus Y]\,(X(\mathcal{R}^\star(\Pi))) = \varnothing \ .$$

Let us count the cases: three binary branching account for $2^3 = 8$ cases. Furthermore, in two cases of the second level, a different operation is permitted, which introduces no further branching, and concludes the generation – as far as $[\mathcal{R} \setminus Y]\,(X(\mathcal{R}^*(\Pi))) = \varnothing$ is concerned. Therefore there are in total $8 + 2 = 10$ cases, of which 2 are concluded. Now, considering the first 8 cases, there remains to generate the emptiness test, with Δ being the set of assumptions generated so far, and writing $\ell = [\mathcal{R} \setminus Y]\,(X(\mathcal{R}^*(\Pi)))$ and $\alpha = [\mathcal{R} \setminus Y]\,(X(\mathcal{R}^\star(\Pi)))$, we have in all cases

$$\varnothing \fatsemi [\ell \rightarrowtail \Delta, \alpha] \fatsemi \Delta \vdash^* \alpha : \mathsf{TA} \vee \Delta \vdash^* \alpha : \mathsf{TA}^= \fatsemi \ell = \varnothing \rightrightarrows \Delta \fatsemi \alpha = \varnothing \ .$$

Because of $\Delta \vdash^* \alpha : \mathsf{TA} \vee \Delta \vdash^* \alpha : \mathsf{TA}^=$, the rule applies in all cases, and thus introduces no further branching nor any new assumptions, preserving the final count of 10 cases. Discarding the paths that are strictly worse than others, there remain only two relevant possibilities: $\langle \mathcal{R} : \mathsf{reg\text{-}pres}^* \rangle$. Indeed, the following steps can be done without introducing any new approximations, for all X and Y. Thus what we have generated so far is a decision procedure if $\mathcal{R} : \mathsf{reg\text{-}pres}^*$, and merely a positive approximation procedure otherwise. Note that this analysis of the cases can easily be automated: for each generated set of assumptions Δ, complete Δ by kind inference into $\Delta' = \Delta \cup \{\, a \mid \Delta \vdash^* a \,\}$; only the cases where Δ' is minimal wrt. inclusion have to be considered. Let us move on to the second part of the rewrite proposition: $X(\mathcal{R}^*(\Pi)) \subseteq Y^{-1}(\mathcal{T})$. The automaton accepting $X(\mathcal{R}^*(\Pi))$ is built exactly as before —again, there are four cases. $Y^{-1}(\mathcal{T})$ is built in one step, and introduces a binary branching:

$$\varnothing \fatsemi Y^{-1}(\mathcal{T}) \rightrightarrows \varnothing, \langle Y : \mathsf{left\text{-}lin} \rangle \fatsemi Y^{-1}(\mathcal{T}) \ .$$

Let us recall the rule for inclusion:

$$\Gamma \fatsemi [\ell \rightarrowtail \Delta, \alpha; \ell' \rightarrowtail \Delta', \alpha'] \fatsemi \Delta' \not\vdash^* \alpha' : + \fatsemi \ell \subseteq \ell' \rightrightarrows \Gamma, \Delta, \Delta' \fatsemi \alpha \subseteq \alpha' \ .$$

Fortunately, $\langle Y : \mathsf{left\text{-}lin} \rangle \not\vdash^* Y^{-1}(\mathcal{T}) : +$, so the rule applies, preserving the two cases. If $\Delta' = \{Y : \mathsf{left\text{-}lin}\}$, then $\Delta' \vdash^* Y^{-1}(\mathcal{T}) : \mathsf{TA}$, and the inclusion test is a positive approximation, as shown by kind inference; if $\Delta' = \varnothing$, then $\Delta' \vdash^* Y^{-1}(\mathcal{T}) : \mathsf{TA}^=$, and it is a decision. Overall, we have a decision procedure if $Y : \mathsf{left\text{-}lin}$ and $\mathcal{R} : \mathsf{reg\text{-}pres}^*$, and a positive approximation procedure otherwise. Formally, a total of 12 cases (theorems) have been generated, though most are redundant. As the procedure generation is easy to understand, for the remaining examples we shall focus on the translation into rewrite propositions.

Our second pattern, $\neg Y \ \wedge \ \Box(\bullet^1 Y \Rightarrow X)$, states that rules of Y may only appear after rules of X —it features a typical case of an implication whose antecedent is in the future wrt. its consequent, and showcases the generality of rule $(X_\hbar)_{[\mathrm{p19}]}$. A four-steps derivation suffices:

$$\updownarrow \frac{\langle \Pi \fatsemi \varepsilon \Vdash \neg Y \ \wedge \ \Box(\bullet^1 Y \Rightarrow X) \rangle}{\updownarrow \dfrac{\langle \Pi \fatsemi \varepsilon \Vdash \neg Y \rangle}{\updownarrow \dfrac{\langle \Pi \fatsemi \star\varepsilon \Vdash \mathcal{R} \setminus Y \rangle\,(X^\star_\varepsilon)}{Y(\Pi) = \varnothing}\,(\neg X)} \ \wedge \ \updownarrow \dfrac{\langle \mathcal{R}^*(\Pi) \fatsemi \varepsilon \Vdash \bullet^1 Y \Rightarrow X \rangle}{\updownarrow \dfrac{\langle \mathcal{R}^*(\Pi) \fatsemi \wr \mathcal{R}, Y \fatsemi \mathcal{R} \mid \overline{\mathbb{N}_2} \int \Vdash X \rangle\,(X_\hbar)}{Y([\mathcal{R} \setminus X]\,(\mathcal{R}^*(\Pi))) = \varnothing \ .}\,(\Rightarrow_\Sigma)}\,(\Box_*)}\,(\wedge)$$

Procedure generation for the rewrite proposition, $Y(\Pi) = \varnothing \ \wedge \ Y([\mathcal{R} \setminus X]\,(\mathcal{R}^*(\Pi))) = \varnothing$, is strictly simpler than for the previous pattern. In this case, the property is decided if $\mathcal{R} : \mathsf{reg\text{-}pres}^*$, and positively approximated otherwise.

The third property pattern, $\Box(X \Rightarrow \circ^1 \Box \neg Y)$, states that any use of a rule in X precludes the subsequent use of any rule in Y, for the remaining execution. Applying the translation rules as usual, we obtain the

following derivation —where $\ell = X(\mathcal{R}^*(\Pi))$:

$$\updownarrow \frac{\langle \Pi \fatsemi \varepsilon \Vdash \Box(X \Rightarrow \circ^1 \Box \neg Y)\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(\Pi) \fatsemi \star\varepsilon \Vdash X \Rightarrow \circ^1 \Box \neg Y\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(\Pi) \fatsemi (\!| X \fatsemi \mathcal{R} \mid \overline{\mathbb{N}}_1 |\!) \Vdash \circ^1 \Box \neg Y\rangle}{\updownarrow \frac{\langle \ell \fatsemi \star\varepsilon \Vdash \Box \neg Y\rangle}{\updownarrow \frac{\langle \ell \fatsemi \star\varepsilon \Vdash \circ^0 \neg Y\rangle}{\updownarrow \frac{\langle \ell \fatsemi \star\varepsilon \Vdash \neg Y\rangle}{\updownarrow \frac{\langle \ell \fatsemi \star\varepsilon \Vdash \mathcal{R} \setminus Y\rangle}{Y\,(X(\mathcal{R}^*(\Pi))) = \varnothing}}}} \wedge \updownarrow \frac{\langle \mathcal{R}(\ell) \fatsemi \varepsilon \Vdash \Box \neg Y\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(\mathcal{R}(\ell)) \fatsemi \star\varepsilon \Vdash \neg Y\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(\mathcal{R}(\ell)) \fatsemi \star\varepsilon \Vdash \mathcal{R} \setminus Y\rangle}{Y\,(\mathcal{R}^*(\mathcal{R}(X(\mathcal{R}^*(\Pi))))) = \varnothing}}}}$$

This, while correct, is not the best possible translation. Indeed, both statements overlap: overall they have the form $Y(\ell) = \varnothing$ and $Y(\mathcal{R}^+(\ell)) = \varnothing$, which would gain to be combined into $Y(\mathcal{R}^*(\ell)) = \varnothing$. This inefficiency can be corrected by noticing that, since $\lambda \models \Box\varphi$, for all formulæ φ, it holds that

$$\langle \Pi \fatsemi \sigma \Vdash \Box\varphi\rangle \iff \langle \Pi \fatsemi \star\sigma \Vdash \Box\varphi\rangle \ .$$

Using this remark, the following nicer translation is obtained immediately; starting again at the fourth step, we apply the "un-starred" version of $(\Box_*)$ instead of the overly general $(\Box_\hbar)$:

$$\updownarrow \frac{\langle X(\mathcal{R}^*(\Pi)) \fatsemi \star\varepsilon \Vdash \Box \neg Y\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(X(\mathcal{R}^*(\Pi))) \fatsemi \star\varepsilon \Vdash \neg Y\rangle}{\updownarrow \frac{\langle \mathcal{R}^*(X(\mathcal{R}^*(\Pi))) \fatsemi \star\varepsilon \Vdash \mathcal{R} \setminus Y\rangle}{Y\,(\mathcal{R}^*(X(\mathcal{R}^*(\Pi)))) = \varnothing}}}$$

The resulting translation can be optimised into $Y\,([\mathcal{R} \setminus Y]^*(X([\mathcal{R} \setminus X]^*(\Pi)))) = \varnothing$, though an easily automatisable process – cf. [6] for a more detailed discussion of this. Ultimately, this property can be decided under the assumptions $\mathcal{R} : \mathsf{reg\text{-}pres}^*$ (or, more precisely, $\mathcal{R} \setminus X : \mathsf{reg\text{-}pres}^*$ and $\mathcal{R} \setminus Y : \mathsf{reg\text{-}pres}^*$), and $X : \mathsf{reg\text{-}pres}$; it is positively approximated otherwise.

6.2. Discussion of Applicability

The applicability of the method is dependent on which temporal formulæ are translatable into rewrite propositions, and on the properties of the rewrite system.

6.2.1. Exactly Translatable Fragment

In this discussion, we shall not consider under-approximated translations at all, leaving all approximations to the procedure. The fragment of LTL which can be translated exactly contains the formulæ generated by the following φ:

$$\begin{aligned} \varphi &:= \top \mid \bot \mid X \mid \neg X \mid \varphi \wedge \varphi \mid \psi \Rightarrow \varphi \mid \bullet^m\varphi \mid \circ^m\varphi \mid \Box\varphi && X \in \wp(\mathcal{R})\,,\ m \in \mathbb{N} \\ \psi &:= \top \mid \bot \mid X \mid \neg X \mid \psi \vee \psi \mid \psi \wedge \psi \mid \bullet^m\psi \mid \circ^m\psi \mid \Box\bullet^m\top \end{aligned}$$

This fragment is obtained from the translations rules, and the necessity that the signature of a formula $\Box\varphi$ appearing in an antecedent be stabilisable to ε, so as to satisfy rule $(X_\hbar)_{[\text{p19}]}$. The form $\Box\bullet^m\top$ is obtained by application of the closed form of $\xi(\Box\varphi)$ obtainable from the proof of Lem. $13_{[\text{p12}]}$. The next section shows what we can do with this fragment.

6.2.2. Coverage of Temporal Specification Patterns

A survey was conducted in [27] on a large number (555) of specifications, from many different sources and application domains. They were classified according to which pattern and scope [as in 28] each specification was an occurrence of. In this section we examine briefly which of those pattern/scope combinations are amenable to checking using our method – often assuming that the pattern atoms P, Q etcetera correspond to

Pattern	Scope					
	Global	Before	After	Between	Until	*Support* $\leqslant$
Absence	**41**	5	12	18	9	48%
Universality	**110**	1	**5**	2	1	96%
Existence	12	1	4	8	1	0%
Bound Existence	0	0	0	1	0	0%
Response	**241**	1	**3**	0	0	99%
Precedence	**25**	0	1	0	0	96%
Resp. Chain	8	0	0	0	0	0%
Prec. Chain	1	0	0	0	0	0%
Support $\leqslant$	95%	0%	32%	0%	0%	83%

Table 1. Partially supported classes of patterns from survey [27]

simple formulæ – and tally the percentages of real-world cases (per the survey) each such combination accounts for. Thus we obtain an upper-bound on the number of cases that fall in the exactly translatable fragment.

Absence patterns are supported for *global* scopes ($\Box\,\neg P$) and *after* scopes ($\Box(Q \Rightarrow \Box\,\neg P)$) —the second derivation of the previous section was an example of that. In both cases, P should follow the grammar $P := X \mid \neg X \mid P \wedge P \mid P \vee P$, so that it always boils down to an atom $P \subseteq \mathcal{R}$ after application of rules $(\wedge_X)_{[\text{p14}]}$, $(\vee_X)$ and $(\neg X)_{[\text{p15}]}$. Note that the latter rule can be applied without caution because, in both cases, P is under the scope of a $\Box$ operator, which, as we have seen in the previous section, renders moot the introduction of a star performed by rule $(\neg X)$. As for Q, it should be an $\mathcal{A}$-LTL formula. The other scopes (*Before*, *Between*, and *Until*) involve heavy uses of $\Diamond$ and $\mathbf{U}$, and thus cannot be translated exactly —in fact we cannot handle those scopes for any pattern.

Universality is very similar to *Absence*, and is handled for *global* scopes ($\Box\,P$) and *after* scopes ($\Box(Q \Rightarrow \Box\,P)$). Again, Q should be $\mathcal{A}$-LTL, but this time – thanks to the absence of negations – there is no particular restriction on P.

Response is partially supported; *global* scopes are of the form $\Box(P \Rightarrow \Diamond\, S)$ and *after* scopes of the form $\Box(Q \Rightarrow \Box(P \Rightarrow \Diamond\, S))$. Although both use a $\Diamond$ operator, this is not strictly needed in some practical cases. For instance, the first formula of the previous section was a response pattern, but instead of stating that there would eventually be a response, it was more specific and used a next operator to assert that the response would take place on the next step. Thus the following response patterns are supported: $\Box(P \Rightarrow \circ^k S)$, $\Box(P \Rightarrow \bullet^k S)$, $\Box(Q \Rightarrow \Box(P \Rightarrow \circ^k S))$ and $\Box(Q \Rightarrow \Box(P \Rightarrow \bullet^k S))$, for $P, Q \in \mathcal{A}$-LTL, and without any specific restriction on S.

Similarly, *Precedence* is partially supported for *global* scopes: $(\Box\,\neg P) \vee (\neg P\,\mathbf{U}\,S)$ is not directly translatable, but whenever the exact number of steps is known, a pattern of the form $\Box(\bullet^k P \Rightarrow S)$ suffices. Again, $P \in \mathcal{A}$-LTL, and S has no special restriction.

The remaining patterns (*Existence*, *Bounded Existence*, *Response Chain* and *Precedence Chain*) are not translatable, and using explicit numbers of steps in those cases would likely be less useful in those cases than it can be for *Response* and *Precedence*

Table 1 is a summary of the results of the survey in [27], that shows to which pattern/scope combination each of the 555 specifications belongs. The combinations which are handled to some extent by an exact translation into rewrite propositions have their numbers shown in bold face. For each pattern, the last cell of the line gives the proportion of cases belonging to a supported (or partially supported) scope. For each scope, the last cell of the column gives the proportion of cases belonging to a supported (or partially supported) pattern. The cell at the bottom-right gives the proportion of cases which belong to a supported (or partially supported) pattern/scope combination, out of all cases which corresponded to a recognisable pattern/scope (511).

It should be kept in mind that the numbers in boldface constitute upper bounds. For instance, it is unknown how many of the global response patterns were formulated – or could have been reformulated – in terms of exact steps, and that is the single largest category. Another way to interpret the table would be as saying that 17% of patterns are clearly not translatable a priori. Nevertheless, the table shows that the patterns and scopes that the method does target are actually the ones which are most likely to matter in practice.

6.2.3. Properties of TRS Encodings

Even if the temporal property under consideration admits of an exact translation into rewrite proposition, there is no guarantee that the corresponding positive approximation procedure will be fine-grained enough to be of any use. How fine or coarse it is depends upon two factors: the depth of the temporal formula, and the properties of the TRS. There are three main constructions which introduce approximations: $\mathcal{R}^*(\ell)$, Π^n_σ, and $\ell \subseteq \ell'$, where ℓ' is not regular. The first kind is almost always present and is well-handled by the existing litterature; the second is avoided by right-linearity and the third by left-linearity. Let us look at some possible application domains:

Java Virtual Machine and Java bytecode semantics are modeled using a TRS in [2], and the translation is implemented for automatic generation from Java bytecode in [19]. All rules are left-linear by construction, and in practice, about 93% of the rules generated by Copster [(l)] are also right-linear. Thus we can expect approximations of the second kind (a single non–right-linear rule suffices), but not of the third.

In [1], the encoding of the Needham–Schroeder protocol is neither right nor left-linear. It is possible to restore linearity properties by abstracting some parts of the TRS – like agents and nonces, as done in the paper – but whether it is better to approximate the TRS to avoid procedure approximations or vice-versa is best determined empirically on a case-by-case basis.

A TRS encoding of CCS specifications without renaming is given in [20]. The rules of the natural encoding are right-linear, but not left-linear. The authors then restore left-linearity in a larger – but finite – TRS. Thus only the first kind of approximation may appear when using the final TRS.

7. Conclusions & Perspectives

The systematic generation of positive approximation procedures to determine whether a given term-rewriting system satisfies a specific linear temporal logic property is addressed in two steps: first, an equivalent rewrite proposition is generated; second, a positive approximation procedure is derived from the rewrite proposition. The first step is achieved through a set of translation rules relying on the notion of signatures, or models of the sub-formulæ appearing as antecedents of the general LTL formula. The second step is served by procedure generation rules, whose main aim is to juggle with the expressive power required to encode the tree languages involved, and to manage approximations. A survey of the existing literature indicates that the method targets widely used patterns of (safety) properties, and that linearity properties which alleviate coarseness in the positive approximation procedure are naturally met by existing TRS models in some problem domains (e.g. bytecode, CCS semantics).

Future works should focus on extending the translatable fragment, in particular concerning operators of the "until" family —though the methods we are currently considering require a more expressive variety of rewrite propositions, possibly involving altered versions of tree automata completion algorithms storing some trace history.

Given the current fragment, however, several more direct extensions are conceivable, as the first translation step is agnostic regarding the details of the rewriting method – thus, provided we have suitable algorithms to calculate the successor image and the set of reachable terms, we can use those in place of the classical ones. For instance, recent works [29] provide a completion algorithm for the innermost rewriting strategy, very suitable for the verification of call-by-value functional programs; with some additional work to get a similar algorithm for one-step image, the second step could yield procedures for innermost rewriting instead of classical rewriting. The same would be true of any other similar rewriting strategy. A related perspective is the integration of equational theories. There already exist versions of tree automata completion parametrised by equations [30], but in those works the equations are used to ensure the precision of the approximation, not to perform reachability analysis modulo rewriting – though it probably can be adapted to such a task, in which case the resulting algorithms could, again, be used in place of the classical ones in the second step. Those perspectives are important for the eventual integration of the verification method in systems such as Maude [21], Tom [31], etc, which mix rewriting with equations and strategies, and therefrom in more specialised rewrite-based verification frameworks such as $\mathbb{K}$, for instance via $\mathbb{K}$-Maude [32].

(l) Figure on the `FactoList2.java` example.

References

[1] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification. In David McAllester, editor, *CADE*, volume 1831 of *LNCS*, pages 271–290. Springer, 2000.

[2] Yohan Boichut, Thomas Genet, Thomas P. Jensen, and Luka Le Roux. Rewriting approximations for fast prototyping of static analyzers. In Baader [33], pages 48–62.

[3] Benoît Boyer and Thomas Genet. Verifying Temporal Regular Properties of Abstractions of Term Rewriting Systems. In *RULE*, volume 21 of *EPTCS*, pages 99–108, 2009.

[4] Roméo Courbis, Pierre-Cyrille Héam, and Olga Kouchnarenko. TAGED Approximations for Temporal Properties Model-Checking. In *CIAA*, volume 5642 of *LNCS*. Springer, 2009.

[5] Pierre-Cyrille Héam, Vincent Hugot, and Olga Kouchnarenko. From linear temporal logic properties to rewrite propositions. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR'12*, volume 7364 of *LNCS*, pages 316–331. Springer, 2012.

[6] Vincent Hugot. *Tree Automata, Approximations, and Constraints for Verification. Tree (Not-Quite) Regular Model-Checking.* These, Université de Franche-Comté, September 2013.

[7] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems - Safety.* Springer, 1995.

[8] Thomas Genet. *Reachability analysis of rewriting for software verification.* Habilitation à diriger des recherches, University of Rennes I, 2009.

[9] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Tree automata with global constraints. In *Developments in Language Theory*, volume 5257 of *LNCS*. Springer, 2008.

[10] Guillaume Feuillade, Thomas Genet, and Valérie Viet Triem Tong. Reachability analysis over term rewriting systems. *J. Autom. Reasoning*, 33(3-4):341–383, 2004.

[11] Yohan Boichut, Pierre-Cyrille Héam, and Olga Kouchnarenko. Approximation-based tree regular model-checking. *Nord. J. Comput.*, 14(3):216–241, 2008.

[12] Santiago Escobar and José Meseguer. Symbolic model checking of infinite-state systems using narrowing. In Baader [33], pages 153–168.

[13] Traian-Florin Serbanuta, Grigore Rosu, and José Meseguer. A rewriting logic approach to operational semantics. *Inf. Comput.*, 207(2):305–340, 2009.

[14] Artur Boronat, Reiko Heckel, and José Meseguer. Rewriting logic semantics and verification of model transformations. In *FASE*, volume 5503 of *LNCS*, pages 18–33. Springer, 2009.

[15] Peter Csaba Ölveczky, editor. *Rewriting Logic and Its Applications - 8th International Workshop, WRLA 2010, Held as a Satellite Event of ETAPS 2010, Paphos, Cyprus, March 20-21, 2010, Revised Selected Papers*, volume 6381 of *Lecture Notes in Computer Science*. Springer, 2010.

[16] José Meseguer. The temporal logic of rewriting: A gentle introduction. In *Concurrency, Graphs and Models*, volume 5065 of *LNCS*. Springer, 2008.

[17] Kyungmin Bae and José Meseguer. The linear temporal logic of rewriting Maude model checker. In Ölveczky [15], pages 208–225.

[18] José Meseguer. Conditioned Rewriting Logic as a United Model of Concurrency. *Theor. Comput. Sci.*, 96(1):73–155, 1992.

[19] Laurent Hubert. Luka Le Roux Nicolas Barré and Thomas Genet. `Copster` homepage. `http://www.irisa.fr/celtique/genet/copster`, 2009.

[20] Roméo Courbis. Rewriting approximations for properties verification over ccs specifications. In Farhad Arbab and Marjan Sirjani, editors, *FSEN*, volume 7141 of *LNCS*, pages 299–315. Springer, 2011.

[21] Steven Eker, José Meseguer, and Ambarish Sridharanarayanan. The Maude LTL model checker and its implementation. In Thomas Ball and Sriram K. Rajamani, editors, *SPIN*, volume 2648 of *LNCS*, pages 230–234. Springer, 2003.

[22] Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In Tobias Nipkow, editor, *RTA*, volume 1379 of *LNCS*, pages 151–165. Springer, 1998.

[23] Toshinori Takai. A verification technique using term rewriting systems and abstract interpretation. In Vincent van Oostrom, editor, *RTA*, volume 3091 of *LNCS*, pages 119–133. Springer, 2004.

[24] Manuel Clavel, Miguel Palomino, and Adrián Riesco. Introducing the itp tool: a tutorial. *J. Univ. Comp. Sci.*, 12(11):1618–1650, 2006.

[25] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 243–320. 1990.

[26] Patrick Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theor. Comput. Sci.*, 277(1-2):47–103, 2002.

[27] Matthew Dwyer, George Avrunin, and James Corbett. Patterns in property specifications for finite-state verification. In *ICSE'99*, pages 411–420. IEEE, 1999.

[28] Matthew Dwyer, George Avrunin, and James Corbett. Property specification patterns for finite-state verification. In *FMSP'98*, pages 7–15. ACM, 1998.

[29] Thomas Genet and Yann Salmon. Reachability Analysis of Innermost Rewriting. Research report, July 2013.

[30] Thomas Genet and Vlad Rusu. Equational approximations for tree automata completion. *J. Symb. Comput.*, 45(5):574–597, 2010.

[31] Moreau Pierre-Etienne, Christophe Ringeissen, and Marian Vittek. A pattern matching compiler for multiple target languages. In *Compiler Construction*, pages 61–76. Springer, 2003.

[32] Traian-Florin Serbanuta and Grigore Rosu. K-maude: A rewriting based tool for semantics of programming languages. In Ölveczky [15], pages 104–122.

[33] Franz Baader, editor. *Term Rewriting and Applications*, volume 4533 of *LNCS*, 2007.