Asynchronous Composition of Local Interface LTL Properties

Alberto Bombardelli[1 [0000−0003−3385−3205]] and
Stefano Tonetta[1 [0000−0001−9091−7899]]

Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Povo TN
{abombardelli,tonettas}@fbk.eu

Abstract. The verification of asynchronous software components is very challenging due to the non-deterministic interleaving of components and concurrent access to shared variables. Compositional approaches decouple the problem of verifying local properties specified over the component interfaces from the problem of composing them to ensure some global property. In this paper, we focus on symbolic model checking techniques for Linear-time Temporal Logic [25] (LTL) properties on asynchronous software components communicating through data ports. Differently from event-based composition, the local properties can specify constraints on the input provided by other components, making their composition more complex.
We propose a new LTL rewriting that translates a local property into a global one taking into account interleaving with other processes. We demonstrate that for every possible global trace, the local LTL property is satisfied by its projection on the local symbols if and only if the rewritten LTL property is satisfied by the global trace. This rewriting is then optimized, reducing the size of the resulting formula and leaving it unchanged when the temporal property is stutter invariant. We also consider an alternative approach where the local formulas are first translated into fair transition systems and then composed. This work has been implemented inside the contract-based design model checking tool OCRA as part of the contract refinement verification suite. Finally, the different composition approaches were compared through an experimental evaluation that covers various types of specifications.

1 Introduction

Software model checking [1,27] is an algorithmic approach used the verification of programs. It combines different methods based on deductive reasoning, abstraction, and state space exploration. Model checking typically specifies the property to be verified in a temporal logic. One of the most common logic used to express properties of programs is first-order Linear-time Temporal Logic (LTL) [25].

A general problem of model checking is the state space explosion problem. The scalability of the method is exacerbated when considering the asynchronous composition of programs, due to the non-deterministic interleaving of components and concurrent access to shared variables. Compositional approaches usually alleviate the problem by decoupling the problem of verifying local properties

specified over the component interfaces from the problem of composing them to ensure some global property. However, the asynchronous composition of local temporal properties may be tricky when considering software components communicating through data ports.

In this paper, we define the asynchronous composition of local LTL properties based on a rewriting $\mathcal{R}_c$ that maps the local constraints on the input/output data of a component c on the global points in which the component is active. In this case, the formulas can be rewritten to take into account interleaving and conjoined with additional constraints ψ_{constr} to encode for example the persistence of variables that are not written by the active process. In this way, it is possible to verify whether a global property ϕ is satisfied by the composition of local properties, by checking the validity of an following LTL formula in the form: $\bigwedge_{c \in C} \mathcal{R}_c(\phi_c) \wedge \psi_{constr} \rightarrow \phi$.

We define the rewriting $\mathcal{R}_c$ for quantifier-free first-order LTL with the "next" operator. In particular, the rewriting of "next", which is important to express input/output properties, needs the use of event-freezing functions, introduced in [29] to relate variables across different time points. We prove that the rewriting is correct, i.e., that for every possible global trace, the local LTL property is satisfied by its projection on the local symbols if and only if the rewritten LTL property is satisfied by the global trace. The main contribution of the paper is an optimized version of the rewriting that takes into account the frame conditions on output data and the stutter invariance of other operators to reduce the size of the resulting formula. We also consider an alternative approach where the local formulas are first translated into fair transition systems and then composed.

The proposed approach has been implemented inside OCRA, which supports a rich extension of LTL and uses a state-of-the-art model checking algorithm implemented in nuXmv [6] as back-ends to check satisfiability. We validated the approach empirically by evaluating the local property and the rewritten one on local traces and their extension with stuttering of local variables. We evaluated the approach on various kind of formulas and components, and compared the different approaches in terms of scalability.

Summarizing, the main contribution of the paper is a rewriting of LTL formulas with the following features:

- it allows to check compositional rules for asynchronous components communicating through input/output data ports;
- it supports compositional reasoning for first-order LTL properties with next and event-freezing functions;
- it is optimized to reduce the size of the resulting formula;
- it has been validated and evaluated on various benchmarks.

The rest of the paper is organized as follows: in Sec. 2, we compare the proposed solution with related works; in Sec. 3, we give some preliminary definitions; in Sec. 4, we formalize the problem; in Sec. 5, we define the rewriting, its optimized version, and the alternative approach based on compilation into transition systems; in Sec. 6, we report on the experimental validation and evaluation; finally, in Sec. 7, we draw the conclusions and some directions for future works.

2 Related works

When dealing with temporal logics such as LTL for asynchronous systems, one of the main references is the work of Leslie Lamport on Temporal Logic of Action (TLA) [17], later enriched with additional operators [18] and to component-based models in [28]. In fact, we adopt the (quantifier-free) first-order version of LTL [20] with the "next" function which is used to specify the succession of actions of a program. TLA natively supports the notion of stuttering for composing asynchronously programs so that the composition is simply obtained by conjoining the specifications. We focus instead on local properties that are specified independently from how the program is composed so that "next" and input/output data refer only to the local execution. To the best of our knowledge, this paper first addresses the asynchronous composition of local first-order LTL properties. In fact, we rewrite "next" terms using the "at next" operator introduced in [29] to take into account interleaving by referring to the value of variables at the next point in time where the component is not stuttering.

As for propositional LTL, the composition of specifications is studied in various papers on assume-guarantee reasoning (see, e.g., [11, 16, 22, 24]) for both synchronous and asynchronous composition. In the case of asynchronous systems, most works focus on fragments of LTL without the next operator, where formulas are always stutter invariant. Other studies investigated how to tackle down state-space explosion for that scenario usually employing techniques such as partial order reduction [4]. However, our work covers a more general setting, where also the presence of input variables makes formulas non stutter invariant.

Similarly to our work, [4] considers a rewriting for LTL with events to map local properties into global ones with stuttering. In [3], a related rewriting is used within an asynchronous version of HyperLTL. However, contrary to this paper, these works do not consider input variables (nor first-order extension) and assume that every variable does not change during stuttering, resulting in a simpler rewriting. In [15], a temporal clock operator is introduced to express properties related to multiple clocks and, in principle, can be used to interpret formulas over the time points in which a component is not stuttering. Its rewriting is indeed similar to the basic version defined in this paper, but is limited to propositional LTL and has not been conceived for asynchronous composition. The optimization that we introduce to exploit the stutter invariance of subformulas results in simpler formulas easy to be analyzed as shown in our experimental evaluation.

The rewriting of asynchronous LTL is similar to the transformation of asynchronous symbolic transition systems into synchronous ones described in [10]. The work considers connections based on events where data are exchanged only upon synchronization (allowing optimizations as in shallow synchronization [5]). Thus, it does not consider components that read from input variables that may be changed by other components. Moreover, [10] is not able to transform temporal logic local properties in global one as in this paper.

3 Background

3.1 Linear temporal logic

In this paper we consider LTL [21] extended with past operators [19] as well as "if-then-else" (ite) and "at next" ($@\tilde{F}$), and "at last" ($@\tilde{P}$) operators from [29]. For simplicity we refer to it simply as LTL.

We work in the setting of Satisfiability Modulo Theory (SMT) [2] and LTL Modulo Theory (see, e.g., [9]). First-order formulas are built as usual by proposition logic connectives, a given set of variables V and a first-order signature Σ, and are interpreted according to a given Σ-theory $\mathcal{T}$. We assume to be given the definition of $M, \mu \models_{\mathcal{T}} \varphi$ where M is a Σ-structure, μ is a value assignment to the variables in V, and φ is a formula. Whenever $\mathcal{T}$ and M are clear from contexts we omit them and simply write $\mu \models \varphi$.

LTL syntax

Definition 1. *Given a signature Σ and a set of variables V, LTL formulas φ are defined by the following syntax:*

$$\varphi := \top | \bot | pred(u_1, \ldots, u_n) | \neg\varphi_1 | \varphi_1 \vee \varphi_2 | X\varphi_1 | \varphi_1 U \varphi_2 | Y\varphi_1 | \varphi_1 S \varphi_2$$

$$u := c | x | func(u_1, \ldots, u_n) | next(u_1) | ite(\varphi, u_1, u_2) | u_1 @\tilde{F}\varphi | u_1 @\tilde{P}\varphi$$

where c, $func$, and $pred$ are respectively a constant, a function, and a predicate of the signature Σ and x is a variable in V.

Apart from $@\tilde{F}$ and $@\tilde{P}$, the operators are standard. $u@\tilde{F}\varphi$ represents the value of u at the next point in time in which φ holds. Similarly, $u@\tilde{P}\varphi$ represents the value of u at the last point in time in which φ holds.

LTL semantic LTL formulas are interpreted over traces, i.e., infinite sequences of assignments to the variables in V. We denote by $\Pi(V)$ the set of all possible traces over the variable set V. Given a trace $\pi = s_0 s_1 \cdots \in \Pi(V)$ and a Σ-structure M, the semantic of a formula φ is defined as follows:

- $\pi, M, i \models pred(u_1, \ldots, u_n)$ iff $pred^M(\pi^M(i)(u_1), \ldots, \pi^M(i)(u_n))$
- $\pi, M, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, M, i \models \varphi_1$ and $\pi, M, i \models \varphi_2$
- $\pi, M, i \models \neg\varphi$ iff $\pi, M, i \not\models \varphi$
- $\pi, M, i \models \varphi_1 U \varphi_2$ iff there exists $k \geq i, \pi, M, k \models \varphi_2$ and for all $l, i \leq l < k, \pi, M, l \models \varphi_1$
- $\pi, M, i \models \varphi_1 S \varphi_2$ iff there exists $k \leq i, \pi, M, k \models \varphi_2$ and for all $l, k < l \leq i, \pi, M, l \models \varphi_1$
- $\pi, M, i \models X\varphi$ iff $\pi, M, i+1 \models \varphi$
- $\pi, M, i \models Y\varphi$ iff $i > 0$ and $\pi, M, i-1 \models \varphi$

where the interpretation of terms $\pi^M(i)$ is defined as follows:

- $\pi^M(i)(c) = c^M$
- $\pi^M(i)(x) = s_i(x)$ if $x \in V$
- $\pi^M(i)(func(u_1, \ldots, u_n)) = func^M(\pi^M(i)(u_1), \ldots, \pi^M(i)(u_n))$
- $\pi^M(i)(next(u)) = \pi^M(i+1)(u)$
- $\pi^M(i)(u@\tilde{F}(\varphi)) = \pi^M(k)(u)$if there exists $k > i$ such that, for all $l, i < l < k, \pi, M, l \not\models \varphi$ and $\pi, M, k \models \varphi$;
  $\pi^M(i)(u@\tilde{F}(\varphi)) = def_{u@\tilde{F}\varphi}$ otherwise.
- $\pi^M(i)(u@\tilde{P}(\varphi)) = \pi^M(k)(u)$if there exists $k < i$ such that, for all $l, i > l > k, \pi, M, l \not\models \varphi$ and $\pi, M, k \models \varphi$;
  $\pi^M(i)(u@\tilde{F}(\varphi)) = def_{u@\tilde{P}\varphi}$ otherwise.
- $\pi^M(i)(ite(\varphi, u_1, u_2)) = \begin{cases} \pi^M(i)(u_1) & \text{if } \pi, M, i \models \varphi \\ \pi^M(i)(u_2) & \text{otherwise} \end{cases}$

and the $pred^M, func^M, c^M$ are the interpretation M of the symbols in Σ, and $def_{u@\tilde{F}\varphi}$ and $def_{u@\tilde{P}\varphi}$ are some default values in domain of M.

Finally, we have that $\pi, M \models \varphi$ iff $\pi, M, 0 \models \varphi$.

In the following, we assume to have a background theory such that the symbols in Σ are interpreted by an implicit structure M (e.g., theory of reals, integers, etc.). We therefore omit M to simplify the notation, writing $\pi, i \models \varphi$ and $\pi(i)(u)$ instead of respectively $\pi, M, i \models \varphi$ and $\pi^M(i)(u)$.

Moreover, we use the following standard abbreviations: $\varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 R \varphi_2 := \neg(\neg\varphi_1 U \neg\varphi_2)$ (φ_1 releases φ_2), $F\varphi := \top U \phi$ (sometime in the future φ), $G\varphi := \neg F \neg\varphi$ (always in the future φ), $O\varphi := \top S \varphi$ (once in the past φ), $H\varphi := \neg O \neg\varphi$ (historically in the past φ), $Z\varphi := \neg Y \neg\varphi$ (yesterday φ or at initial state), $X^n\varphi := XX^{n-1}\varphi$ with $X^0\varphi := \varphi$, $Y^n\varphi := YY^{n-1}\varphi$ with $Y^0\varphi := \varphi$, $Z^n\varphi := ZZ^{n-1}\varphi$ with $Z^0\varphi := \varphi$, $F^{\leq n}\varphi := \varphi \vee X\varphi \vee \cdots \vee X^n\varphi$, $G^{\leq n}\varphi := \varphi \wedge X\varphi \wedge \cdots \wedge X^n\varphi$, $O^{\leq n}\varphi := \varphi \vee Y\varphi \vee \cdots \vee Y^n\phi$, $H^{\leq n}\varphi := \varphi \wedge Z\varphi \wedge \cdots \wedge Z^n\varphi$.

Since this paper heavily relies on the release operator, we explicitly define its semantics as follows:

$\pi, M, i \models \varphi_1 R \varphi_2$ iff for all $l \geq i$, $\pi, M, l \models \varphi_2$ or there exists $k \geq i$, $\pi, M, k \models \varphi_1$ and for all $i \leq l' \leq k$, $\pi, M, l' \models \varphi_2$

3.2 Interface transition systems

In this paper, we represents I/O components as Interface Transition Systems, a symbolic version of interface automata [13] that considers I/O variables instead of I/O actions.

Definition 2. *An Interface Transition System (ITS) $\mathcal{M}$ is a tuple $\mathcal{M} = \langle V_I, V_O, V_H, \mathcal{I}, \mathcal{T}, \mathcal{F} \rangle$ where:*

- *V_I is the set of input variables, V_O is the set of output variables, V_H is the set of internal variables where $V_I \cap V_O = \emptyset$, $V_I \cap V_H = \emptyset$ and $V_O \cap V_H = \emptyset$.*
- *$V := V_I \cup V_O \cup V_H$ denotes the set of the variables of $\mathcal{M}$*

- $\mathcal{I}$ *is the initial condition, a formula over* $V_O \cup V_H$,
- $\mathcal{T}$ *is the transition condition, a formula over* $V \cup V'_O \cup V'_H$ *where* V'_O *and* V'_H *are respectively the primed versions of* V_O *and* V_H
- $\mathcal{F}$ *is the set of fairness constraints, a set of formulas over* V.

A symbolic transition system $\mathcal{M} = \langle V, S, \mathcal{I}, \mathcal{T}, \mathcal{F} \rangle$ *is an interface transition system without input/output variables (i.e.,* $\langle \emptyset, \emptyset, V, S, \mathcal{I}, \mathcal{T}, \mathcal{F} \rangle$*).*

Definition 3. *A trace* π *of an ITS* $\mathcal{M}$ *is a trace* $\pi = s_0 s_1 s_2 \cdots \in \Pi(V)$ *such that* $s_0 \models \mathcal{I}$*, for all* i*,* $s_i \cup s'_{i+1} \models \mathcal{T}$*, and for all* $f \in \mathcal{F}$*, for all* i*, there exists* $j > i$*,* $s_j \models f$*. The language* $\mathscr{L}(\mathcal{M})$ *of an interface transition system* $\mathcal{M}$ *is the set of all traces of* $\mathcal{M}$*. Given an LTL formula* φ*,* $\mathcal{M} \models \varphi$ *iff, for all traces* π *of* $\mathcal{M}$*,* $\pi \models \varphi$*.*

The asynchronous composition of two ITS is an ITS where the transitions of the two original ITS occurs concurrently. To compose two interface transition systems, their variables must be compatible.

Definition 4. *Two ITS* $\mathcal{M}_1, \mathcal{M}_2$ *are compatible iff they share respectively only input with output (i.e.* $V^1 \cap V^2 = (V^1_O \cap V^2_I) \cup (V^1_I \cap V^2_O)$*)*

The asynchronous composition of ITS should allow certain ITS to run their transitions while the other transition systems freeze. To encode this behaviour symbolically, the composition adds one *stuttering* variable for each interface transition system. A stuttering variable is a Boolean variable that tells whether a specific component is frozen or if it is executing its transition. We denote $st^{\mathcal{M}}$ as the stuttering variable of the ITS $\mathcal{M}$.

Moreover, we introduce new transition conditions $\psi^{\mathcal{M}}_{cond}$ to express the fact that an ITS $\mathcal{M}$ inside a composition do not change their output and internal variables when their stuttering variables are true. Formally, for all ITS $\mathcal{M}$: $\psi^{\mathcal{M}}_{cond} = st^{\mathcal{M}} \rightarrow \bigwedge_{v \in V_O \cup V_H} (v = v')$

Definition 5. *Let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two compatible interface transition systems.* $\mathcal{M}_1 \otimes \mathcal{M}_2 = \langle V_I, V_O, V_H, \mathcal{I}, \mathcal{T}, \mathcal{F} \rangle$ *where:*

- $V_I = V^1_I \cup V^2_I \setminus ((V^1_I \cap V^2_O) \cup (V^1_O \cap V^2_I))$
- $V_O = V^1_O \cup V^2_O \setminus ((V^1_I \cap V^2_O) \cup (V^1_O \cap V^2_I))$
- $V_H = V^1_H \cup V^2_H \cup \{st^{\mathcal{M}_1}, st^{\mathcal{M}_2}\} \cup ((V^1_I \cap V^2_O) \cup (V^1_O \cap V^2_I))$
- $\mathcal{I} = \mathcal{I}^1 \wedge \mathcal{I}^2$
- $\mathcal{T} = (\neg st^{\mathcal{M}_1} \rightarrow \mathcal{T}^1) \wedge (\neg st^{\mathcal{M}_2} \rightarrow \mathcal{T}^2) \wedge \psi^{\mathcal{M}_1}_{cond} \wedge \psi^{\mathcal{M}_2}_{cond}$
- $\mathcal{F} = \{\neg st^{\mathcal{M}_1}, \neg st^{\mathcal{M}_2}\} \cup \{\varphi^1 \wedge \neg st^{\mathcal{M}_1} | \varphi^1 \in \mathcal{F}^1\} \cup \{\varphi^2 \wedge \neg st^{\mathcal{M}_2} | \varphi^2 \in \mathcal{F}^2\}$

The definition can be easily generalized to n ITSs $\mathcal{M}_1 \otimes ... \otimes \mathcal{M}_n$

Definition 6. *Let* $\mathcal{M} = \mathcal{M}_1 \otimes \mathcal{M}_2$ *be the asynchronous composition of two ITS* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*, let* $\pi = s_0 s_1 \ldots$ *be a trace of* $\mathcal{M}$*. A pair of consecutive assignments to states* s_i, s_{i+1} *of* π *is called stuttering transition w.r.t.* $\mathcal{M}_1$*,* $\mathcal{M}_2$ *iff* $s_i \models st^{\mathcal{M}_1}, st^{\mathcal{M}_2}$ *respectively.*

Definition 7. *Let $\pi = s_0 s_1 \dots$ be a trace of an ITS $\mathcal{M}$ and $V' \subseteq V$ a set of symbols of $\mathcal{M}$. We denote $s_i(V')$ as the restriction of the assignment s_i to the symbols of V'; moreover, we denote $\pi_{|V'} := s_0(V')s_1(V')\dots$ as the restriction of all the state assignments of π to the symbols of V'. Furthermore, we denote $\mathscr{L}(\mathcal{M})_{|V'} = \{\pi_{|V'} | \pi \in \mathscr{L}(\mathcal{M})\}$ as the restriction of all the traces of the language of an ITS $\mathcal{M}$ to a set of symbols $V' \subseteq V$.*

Definition 8. *Let $\pi = s_0 s_1 \dots$ be a trace of the asynchronous composition of n ITS $\mathcal{M}_1, \dots, \mathcal{M}_n$. By fairness constraints on stuttering there are infinitely many points $i_0, i_1, \dots$ such that for all $j : \pi, i_j \models \neg st^{\mathcal{M}_h}$ and for all $k, i_j < k < i_{j+1}.\pi, k \models st^{\mathcal{M}_h}$. We define the projection of trace π over a component $\mathcal{M}_h$ as follow:*

$$Pr_{\mathcal{M}_h}(\pi) = s_{i_0}(V_h), s_{i_1}(V_h) \dots$$

Definition 9. *Let $\mathcal{M} = \mathcal{M}_1 \otimes \dots \otimes \mathcal{M}_n$, let π be a trace of $\mathcal{M}_h$. We define the inverse operator of Pr, denoted by Pr^{-1}.*

$$Pr^{-1}_{\mathcal{M}_h}(\pi) = \{\pi' | Pr_{\mathcal{M}_h}(\pi') = \pi\}$$

4 Formal problem

4.1 Asynchronous composition of properties of ITS

Compositional verification proves the properties of a system by proving the local properties on components and by checking that the composition of the local properties satisfy the global one (see [26] for a generic overview). This reasoning is expressed formally by inference 1, which is parametrized by a function γ_S that combines the components' implementations and a related function γ_P that combines the local properties.

Inference 1 *Let $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_n$ be a set of n components, $\varphi_1, \varphi_2, \dots, \varphi_n$ be local properties on each component, γ_S is a function that defines the composition of $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_n$, γ_P combines the properties depending on the composition of γ_S and φ a property. The following inference is true:*

$$\frac{\dfrac{\mathcal{M}_1 \models \varphi_1, \mathcal{M}_2 \models \varphi_2, \dots, \mathcal{M}_n \models \varphi_n}{\gamma_S(\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_n) \models \gamma_P(\varphi_1, \varphi_2, \dots, \varphi_n)} \quad \gamma_P(\varphi_1, \varphi_2, \dots, \varphi_n) \models \varphi}{\gamma_S(\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_n) \models \varphi}$$

In our setting, the components $\mathcal{M}_1, \dots, \mathcal{M}_n$ are represented by ITSs (see definition 2) and γ_S is defined as the generalization of the asynchronous composition of definition 5 for n ITS: $\gamma_S(\mathcal{M}_1, \dots \mathcal{M}_n) = \mathcal{M}_1 \otimes \dots \otimes \mathcal{M}_n$.

The problem we address in this paper is to define γ_P such that the above inference rule is correct. In order to asynchronously combine the local properties, each property must be rewritten considering stuttering transitions and, evaluating input variables only in active transitions. Formally, we want that for all trace π of $\gamma_S(\mathcal{M}_1, \dots, \mathcal{M}_n), Pr_{\mathcal{M}_1}(\pi) \models \varphi_1 \wedge \dots \wedge Pr_{\mathcal{M}_n}(\pi) \models \varphi_n \Leftrightarrow \pi \models \gamma_P(\varphi_1, \dots, \varphi_n)$.

Thus, we require a rewriting function that maps local properties into their global counterparts. This requirement is expressed as follows. For each trace π of an ITS $\mathcal{M}$, for each global trace $\pi^{ST} \in Pr^{-1}(\pi)$: $\pi \models \varphi$ iff $\pi^{ST} \models \mathcal{R}^*(\varphi)$ where φ is a local LTL property in the language of $\mathcal{M}$. As for the event based TS, we need some conditions Ψ_{cond} that we call *frame condition* to guarantee persistency of output variables and to guarantee fairness on components activity. $\Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n) := \psi_{cond}^{\mathcal{M}_1} \wedge \cdots \wedge \psi_{cond}^{\mathcal{M}_n} \wedge GF\neg st^{\mathcal{M}_1} \wedge \cdots \wedge GF\neg st^{\mathcal{M}_n}$ where ψ_{cond} is from definition 5. The final result in this case would be

$$\gamma_P := \mathcal{R}^*(\varphi_1) \wedge \cdots \wedge \mathcal{R}^*(\varphi_n) \wedge \Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n)$$

Example 1. Let $\mathcal{M}_1$ be an ITS with c_2 as input variable, c_1 as output variable and $\varphi_1 : c_1 = 0 \wedge G((c_1 < c_2 \wedge c_1' = c_1 + 1) \vee (c_1 \geq c_2 \wedge c_1' = c_1))$ as its local property. Let $\mathcal{M}_2$ be another ITS with c_1 as input variable, c_2 as output variable, p as parameter and $\varphi_2 : c_2 = p \wedge G((c_2' = c_2 - 1)U(c_2 = 0 \wedge c_2' = c_1))$ as its local property. Suppose that we want to prove that the composition of the two properties satisfies the *global* property $\varphi : GF(c_1 = c_1')$. To check if φ holds we check the validity of $\mathcal{R}^*_{\mathcal{M}_1}(\varphi_1) \wedge \mathcal{R}^*_{\mathcal{M}_2}(\varphi_2) \wedge \Psi_{cond}(\mathcal{M}_1, \mathcal{M}_2) \rightarrow \varphi$.

In this example, c_1 is increased only when c_1 is lower than c_2. When considering asynchronous composition, c_2 might change while $\mathcal{M}_1$ is stuttering. In this case, the challenge in finding a correct $\mathcal{R}^*$ is that since c_2 might change while $\mathcal{M}_1$ is stuttering, then the rewriting must evaluate c_2 only when $\mathcal{M}_1$ is active.

4.2 Asynchronous composition of properties of event based TS

For completeness, we compare the problem defined above with the case of a event-based asynchronous composition, where the transition systems run concurrently with only shared events used for synchronization. If we consider the asynchronous composition of event based TS to represent the function γ_S, we can use the LTL rewriting function T defined in [4] on each $\varphi_1, \ldots, \varphi_n$ inside γ_P to compose the properties. T simply rewrites events and X operators and leaves the other parts of formulas unchanged. We apply T to each φ_i, then, we put these rewritten properties in conjunction with a constraint Ψ that ensures that variables do not change during stuttering transitions and that events do not occur during stuttering transition. $\Psi = \bigwedge_{1 \leq i \leq n}(G(st^{\mathcal{M}_i} \rightarrow \bigwedge_{v \in V^i} v = v' \wedge \bigwedge_{e \in E^i}))$ where V^i and E^i are the sets of respectively variables and events of each $\mathcal{M}^i$. Finally, $\gamma_P(\mathcal{M}_1, \ldots, \mathcal{M}_n) = T(\varphi_1) \wedge \cdots \wedge T(\varphi_n) \wedge \Psi$ In this case, the composition is limited to components with synchronous event communications. Thus, no input variable that is updated by other components can be considered in this model of composition.

5 Rewriting [1]

This section contains the main contributions of this paper. First, a rewriting $\mathcal{R}^*_{\mathcal{M}}$ that transform local LTL properties into their global counterparts. Second, an optimised version of $\mathcal{R}^*_{\mathcal{M}}$, ${\mathcal{R}^{\theta}}^*_{\mathcal{M}}$, which exploits the concept of *stutter tolerance* (see definition 14) to reduce the size of the generates formula. Finally, an alternative approach that transforms the local LTL formulas into ITS and then composes the ITS asynchronously.

We introduce the *map* function; a function that maps the position of a state in a local trace to its position in a global trace.

To simplify the notation, we assume to be given an ITS $\mathcal{M}$, a trace π of $\mathcal{M}$, a local property φ and a local term u. For brevity, we refer to $map^{st}_{\pi^{ST}}$,$\mathcal{R}_{\mathcal{M}}$, $\mathcal{R}^*_{\mathcal{M}}$, $\mathcal{R}^{\theta}_{\mathcal{M}}$, ${\mathcal{R}^{\theta}}^*_{\mathcal{M}}$, $Pr^{-1}_{\mathcal{M}}$ and $st^{\mathcal{M}}$ as respectively $map_{\pi^{ST}}$, $\mathcal{R}$, $\mathcal{R}^*$, $\mathcal{R}^{\theta}$, ${\mathcal{R}^{\theta}}^*$,Pr^{-1} and st.

Definition 10. *For all $\pi^{ST} \in Pr^{-1}(\pi)$, for all $k \in \mathbb{N} : \neg st^{\pi^{ST}}_{occ}(k) := j$ s. t. $\pi^{ST}, j \models \neg st$ and for all $k \leq l < j : \pi^{ST}, l \models st$. $\neg st^{\pi^{ST}}_{occ}(k)$ denotes the position of the first occurrence of $\neg st$ from point k. We also define map as follows: For all i:*

$$map_{\pi^{ST}}(i) := \begin{cases} \neg st^{\pi^{ST}}_{occ}(0) & \text{if } i = 0 \\ \neg st^{\pi^{ST}}_{occ}(map_{\pi^{ST}}(i-1)+1) & \text{if } i > 0 \end{cases}$$

5.1 $\mathcal{R}$ rewriting

As we mentioned in section 4.1, we want a rewriting that is able to map each local property φ into its global counterpart. In this case, each global property must be satisfied in $Pr^{-1}(\pi)$ iff φ is satisfied in π. We start by proposing a rewriting that maps an LTL formula to another formula such that the augmented traces satisfy the rewritten formula in the active transitions if and only if the original traces satisfy them in the same transitions.

Definition 11. *We define $\mathcal{R}$ as the following rewriting function:*

1. $\mathcal{R}(a) := a$
2. $\mathcal{R}(\varphi \vee \psi) := \mathcal{R}(\varphi) \vee \mathcal{R}(\psi)$
3. $\mathcal{R}(\neg\varphi) := \neg\mathcal{R}(\varphi)$
4. $\mathcal{R}(X\psi) := X(\neg stR(st \vee \mathcal{R}(\psi)))$
5. $\mathcal{R}(\varphi U \psi) := (st \vee \mathcal{R}(\varphi))U(\neg st \wedge \mathcal{R}(\psi))$
6. $\mathcal{R}(Y\varphi) := Y(stS(\neg st \wedge \mathcal{R}(\varphi)))$
7. $\mathcal{R}(\varphi S \psi) := (st \vee \mathcal{R}(\varphi))S(\neg st \wedge \mathcal{R}(\psi))$
8. $\mathcal{R}(func(\psi_1, ..., \psi_n)) := func(\mathcal{R}(\psi_1), ..., \mathcal{R}(\psi_n))$
9. $\mathcal{R}(pred(\psi_1, ..., \psi_n)) := pred(\mathcal{R}(\psi_1), ..., \mathcal{R}(\psi_n))$
10. $\mathcal{R}(ite(\psi, \psi_1, \psi_2)) := ite(\mathcal{R}(\psi), \mathcal{R}(\psi_1)\mathcal{R}(\psi_2))$

[1] The proofs of the theorems and lemmas of this section can be found in the appendix of the completed version of the paper at: https://es-static.fbk.eu/people/bombardelli/papers/nfm22/nfm-extended.pdf

11. $\mathcal{R}(next(\psi) := \psi@\tilde{F}\neg st$
12. $\mathcal{R}(\psi@\tilde{F}\psi_1) := \mathcal{R}(\psi)@\tilde{F}(\mathcal{R}(\psi_1) \wedge \neg st)$
13. $\mathcal{R}(\psi@\tilde{P}\psi_1) := \mathcal{R}(\psi)@\tilde{P}(\mathcal{R}(\psi_1) \wedge \neg st)$

The property of $\mathcal{R}$ is defined in the following lemma:

Lemma 1 *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi)$, for all i:*

$$\pi^{ST}, i \models \varphi \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(i) \models \mathcal{R}(\varphi) \quad \pi(i)(u) = \pi^{ST}(map_{\pi^{ST}}(i))(\mathcal{R}(u))$$

Lemma 1 shows that $\mathcal{R}$ guarantees that satisfiability is preserved in the active transitions of the global traces. However, $map_{\pi^{ST}}(0)$ is not always granted to be equal to 0 (see definition 9), and thus, we need to find a rewriting that guarantees that satisfiability is preserved also in the first transition.

Definition 12. *We define $\mathcal{R}^*$ as $\mathcal{R}^*(\varphi) := \neg stR(st \vee \mathcal{R}(\varphi))$*

Lemma 2 *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi) : \pi^{ST}, map_{\pi^{ST}}(0) \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST}, 0 \models \mathcal{R}^*(\varphi))$*

Using lemma 1 and lemma 2 together we obtain the following theorem:

Theorem 1. *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi) : \pi \models \varphi \Leftrightarrow \pi^{ST} \models \mathcal{R}^*(\varphi)$*

Theorem 1 shows that $\mathcal{R}^*$ is able to translate a local LTL property into a global property without changing its semantics in term of traces. Using $\mathcal{R}^*$ is possible to transform local properties with I/O variables.

Definition 13. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be n ITS and $\varphi_1, \ldots, \varphi_n$ be LTL formulas on the language of each $\mathcal{M}_i$. We define $\gamma_P(\varphi_1, \ldots, \varphi_n) = \mathcal{R}^*_{\mathcal{M}_1}(\varphi_1) \wedge \cdots \wedge \mathcal{R}^*_{\mathcal{M}_n}(\varphi_n) \wedge \Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n)$*

Corollary 1 *Using γ_P from definition 13, γ_S from section 4.1, for all compatible ITS $\mathcal{M}_1, \ldots, \mathcal{M}_n$, for all local properties $\varphi_1, \ldots, \varphi_n$ over the language of respectively $\mathcal{M}_1, \ldots, \mathcal{M}_n$, for all global properties φ: Inference 1 holds.*

Example 2. Consider the specifications of example 1. Through $\mathcal{R}^*$ we can define the asynchronous parallel composition of φ_1 and φ_2:

- $\mathcal{R}^*_{\mathcal{M}_1}(\varphi_1) : \neg st^{\mathcal{M}_1} R(st \vee (c_1 = 0 \wedge G(st^{\mathcal{M}_1} \vee (c_1 < c_2 \wedge c_1@\tilde{F}\neg st^{\mathcal{M}_1} = c1 + 1 \vee c_1 \geq c_2 \wedge c_1@\tilde{F}\neg st^{\mathcal{M}_1} = c_1)))$
- $\mathcal{R}^*_{\mathcal{M}_2}(\varphi_2) : \neg st^{\mathcal{M}_2} R(st^{\mathcal{M}_2} \vee c_2 = p \wedge G(st^{\mathcal{M}_2} \vee ((st^{\mathcal{M}_2} \vee c_2@\tilde{F}\neg st^{\mathcal{M}_2} = c_2 - 1)U(\neg st^{\mathcal{M}_2} \wedge c_2 = 0 \wedge c_2@\tilde{F}\neg st^{\mathcal{M}_2} = c_1))))$
- $\Psi_{cond}(\mathcal{M}_1, \mathcal{M}_2) = G(\neg st^{\mathcal{M}_1} \vee c_1 = c_1') \wedge GF\neg st^{\mathcal{M}_1} \wedge G(\neg st^{\mathcal{M}_2} \vee c_2 = c_2') \wedge GF\neg st^{\mathcal{M}_2}$

Each next operator is rewritten as an at next ($@\tilde{F}$). The intuition is that we want to evaluate the variable only in the next transition that does not stutter. c_1 and c_2 are evaluated at the first non stuttering transition, the intuition is that the local initial state is not necessary the global initial state. Finally, using γ_P we can compose φ_1 and φ_2 asynchronously permitting us to check whether or not φ holds. It should be noted that the correctness of the rewriting is guaranteed also removing the constraints on output variables, however this constraint is desirable since it guarantees persistence of data which is a rather realistic property.

5.2 Optimization

The rewriting $\mathcal{R}^*$ is general and works for all the LTL formulas. However, this rewriting increases the size of the formula and consequently, the time required to verify the final specification. There are common cases where it is not necessary to rewrite part of the specification. For example $GF\varphi$ is rewritten as $G(st \vee F(\neg st \wedge \mathcal{R}(\varphi)))$ while it could be rewritten as $GF(\neg st \wedge \mathcal{R}(\varphi))$ (as for fairness constraints). X of a local variable $X\varphi$ is rewritten as $X(\neg st R \mathcal{R}(\varphi))$ while by Ψ_{cond} of section 4 it could remain unchanged. This section identifies, formalizes and demonstrates the cases where such optimization can be applied.

We introduce the concept of *stutter-tolerance*. A formula is said *stutter-tolerant* if it keeps the same value when rewritten through $\mathcal{R}$ in all consecutive stuttering transitions.

Definition 14. *An LTL formula φ is said stutter-tolerant w.r.t. $\mathcal{R}$ iff:*
For all π, for all $\pi^{ST} \in Pr^{-1}(\pi)$, for all i : for all $map_{\pi^{ST}}(i-1) < j < map_{\pi^{ST}}(i)$:

$$\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(i) \models \mathcal{R}(\varphi)$$

Lemma 3 *Until, yesterday and at last formulas are stutter-tolerant w.r.t. $\mathcal{R}$*

Definition 15. *An LTL formula φ is syntactically stutter-tolerant iff one of the following condition holds:*

- *φ is an until formula or a yesterday formula or an at last formula*
- *$\varphi = \psi_1 \vee \psi_2$ and ψ_1 and ψ_2 are syntactically stutter-tolerant*
- *$\varphi = \neg\psi$ and ψ is syntactically stutter-tolerant*
- *$\varphi = s$ and $s \in V_O \cup V_H$*

Lemma 4 *Syntactically stutter-tolerant formulas are stutter-tolerant w.r.t $\mathcal{R}$*

Using the notion of syntactically stutter-tolerant formula, we define a new optimized rewriting. If the sub-formulas of φ are syntactically stutter-tolerant, then the φ is not rewritten according to $\mathcal{R}$. To demonstrate the correctness of the rewriting, we provide two lemmas that construct the main theorem.

Definition 16. *We define $\mathcal{R}^\theta$ as follows:*

1. $\mathcal{R}^\theta(s) = \mathcal{R}(s)$ *if* $s \in V$
2. $\mathcal{R}^\theta(\varphi \vee \psi) = \mathcal{R}^\theta(\varphi) \vee \mathcal{R}^\theta(\psi)$
3. $\mathcal{R}^\theta(\neg\varphi) = \neg\mathcal{R}^\theta(\varphi)$
4. $\mathcal{R}^\theta(X\psi) = \begin{cases} X(\mathcal{R}^\theta(\psi)) & \text{if } \psi \text{ is synt. st.tol.} \\ X(\neg st R(st \vee \mathcal{R}^\theta(\psi))) & \text{otherwise} \end{cases}$
5. $\mathcal{R}^\theta(\varphi U \psi) = \begin{cases} \mathcal{R}^\theta(\varphi) U \mathcal{R}^\theta(\psi) & \text{if } \psi \text{ is synt. st.tol.} \\ (st \vee \mathcal{R}^\theta(\varphi)) U (\neg st \wedge \mathcal{R}^\theta(\psi)) & \text{otherwise} \end{cases}$
6. $\mathcal{R}^\theta(Y\psi) = Y(stS(\neg st \wedge \mathcal{R}^\theta(\psi)))$
7. $\mathcal{R}^\theta(\varphi S \psi) = \begin{cases} \mathcal{R}^\theta(\varphi) S \mathcal{R}^\theta(\psi) & \text{if } \psi \text{ is synt. st.tol} \\ (st \vee \mathcal{R}^\theta(\varphi)) S (\neg st \wedge \mathcal{R}^\theta(\psi)) & \text{otherwise} \end{cases}$

8. $\mathcal{R}^\theta(func(\psi_1, ..., \psi_n)) = func(\mathcal{R}^\theta(\psi_1), ..., \mathcal{R}^\theta(\psi_n))$
9. $\mathcal{R}^\theta(pred(\psi_1, ..., \psi_n)) = pred(\mathcal{R}^\theta(\psi_1), ..., \mathcal{R}^\theta(\psi_n))$
10. $\mathcal{R}^\theta(ite(\psi, \psi_1, \psi_2)) = ite(\mathcal{R}^\theta(\psi), \mathcal{R}^\theta(\psi_1), \mathcal{R}^\theta(\psi_2))$
11. $\mathcal{R}^\theta(next(\psi)) = \begin{cases} next(\mathcal{R}^\theta(\psi)) & \textit{if } \psi \textit{ is synt. st.tol.} \\ \mathcal{R}^\theta(\psi)@F\neg st & \textit{otherwise} \end{cases}$
12. $\mathcal{R}^\theta(\psi@F\psi_1) = \begin{cases} \mathcal{R}^\theta(\psi)@F\mathcal{R}^\theta(\psi_1) & \textit{if } \psi \textit{ is synt. st. tol.} \\ \mathcal{R}^\theta(\psi)@F(\neg st \wedge \mathcal{R}^\theta(\psi_1)) & \textit{otherwise} \end{cases}$
13. $\mathcal{R}^\theta(\psi@\tilde{P}\psi_1) = \mathcal{R}^\theta(\psi)@\tilde{P}(\neg st \wedge \mathcal{R}^\theta(\psi_1))$

Lemma 5 *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi)$, for all i:*

$$\pi, i \models \varphi \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(i) \models \mathcal{R}^\theta(\varphi) \quad \pi(i)(u) = \pi^{ST}(map_{\pi^{ST}}(i))(\mathcal{R}^\theta(u))$$

Definition 17. *We define $\mathcal{R}^{\theta^*}$ as follows:*

$$\mathcal{R}^{\theta^*}(\varphi) := \begin{cases} \mathcal{R}^\theta(\varphi) & \textit{if } \varphi \textit{ is synt. st.tol.} \\ \neg stR(st \vee \mathcal{R}^\theta(\varphi)) & \textit{otherwise} \end{cases}$$

Lemma 6 *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi) : \pi^{ST}, map_{\pi^{ST}}(0) \models \mathcal{R}^\theta(\varphi) \Leftrightarrow \pi^{ST}, 0 \models \mathcal{R}^{\theta^*}(\varphi)$*

Theorem 2. *For all π, for all $\pi^{ST} \in Pr^{-1}(\pi) : \pi \models \varphi \Leftrightarrow \pi^{ST} \models \mathcal{R}^{\theta^*}(\varphi)$*

Example 3. Consider the specifications of example 1. As for example 2 we can define the asynchronous parallel composition of φ_1 and φ_2 using $\mathcal{R}^{\theta^*}$:

- $\mathcal{R}^{\theta^*}_{\mathcal{M}_1}(\varphi_1) : c_1 = 0 \wedge G(st^{\mathcal{M}_1} \vee (c_1 < c_2 \wedge c_1' = c1 + 1 \vee c_1 \geq c_2 \wedge c_1' = c_1)))$
- $\mathcal{R}^{\theta^*}_{\mathcal{M}_2}(\varphi_2) : c_2 = p \wedge G((st^{\mathcal{M}_2} \vee c_2' = c_2 - 1)U(\neg st^{\mathcal{M}_2} \wedge c_2 = 0 \wedge c_2' = c_1))$

This example shows how much the optimization can reduce the size of the formula. Since $c_1 = 0$ is an output formula and since G is an until operator, $\mathcal{R}^{\theta^*}$ removes the initial $\neg stR(st \vee \mathcal{R}^\theta(\varphi)$. Furthermore, thanks to Ψ_{cond}, both next expressions can be optimized. Another applied optimization is that the rewriting of φ_2 does not need to add stuttering disjunction on G since until is a syntactically stutter formula. However, since φ_1 and φ_2 are not stutter invariant formulas, both specifications are partially modified by $\mathcal{R}^{\theta^*}$. In particular, inside φ_1 $\mathcal{R}^{\theta^*}$ applies the rewriting of G since next formulas are not stutter tolerant, the same happens with φ_2 where U is rewritten according to $\mathcal{R}$.

5.3 Alternative approach for asynchronous composition

In this section, we consider an alternative approach based on the asynchronous composition of ITS. We exploit the transformation from LTL formula to transition system of [12] to generate ITS to be asynchronously composed. ITS have limited expressibility for initial and transition conditions (see definition 2). Initial conditions cannot refer to input formula while transition conditions cannot

refer to next input formulas. Since LTL does not suffer from this limitation, it is necessary to adapt the ITS construction to fully express all possible LTL properties. Thus, we introduce internal variables that mimic the values of the input variables at each transition; exploiting the asynchronous composition, during stuttering transitions these variables will *guess* the value of the input variables at the next occurrence of not stutter.

Definition 18. *Let $\mathcal{M}$ be an ITS and let φ be an LTL formula over its symbols. We define $LTL2IntTS(\mathcal{M}, \varphi) := \langle V_I, V_O, {V_H}', \mathcal{I}, \mathcal{T}, \mathcal{F}_\varphi \rangle$ where:*

- *$LTL2TS(\varphi) = \langle V_\varphi, \mathcal{I}_\varphi, \mathcal{T}_\varphi, \mathcal{F}_\varphi \rangle$ is the transition system generated from φ*
- *${V_H}' = V_H \cup V^{guess} \cup (V_\varphi \setminus V)$*
- *$\mathcal{I} = \mathcal{I}_\varphi \lceil V_I / V^{guess} \rfloor$*
- *$\mathcal{T} = \mathcal{T}_\varphi \lceil {V_I}' / V^{guess'} \rfloor \wedge \bigwedge_{\bar{v} \in V^{guess}} (\bar{v} = v)$*
- *$V^{guess} = \{\bar{v} | v \in V_I\}$ where each $\bar{v}$ is a copy of each v*

Lemma 7 *Let $\mathcal{M}$ be an ITS, let φ be an LTL property over the language of $\mathcal{M}$. $\mathcal{M}_\varphi = LTL2IntTS(\mathcal{M}, \varphi)$ is a valid ITS and $\mathcal{M}_\varphi \models \varphi$*

Lemma 7 ensures that $LTL2IntTS$ generates an ITS that satisfy the property φ. Thus, using $LTL2IntTS$ with the asynchronous composition of ITS of definition 5 we generate the composed ITS.

The remainder of this section demonstrates the equivalence between the this approach with the rewriting techniques. The following lemma ensures that a trace π is part of the language of the composition of the ITS defined by $LTL2IntTS$ if and only if the projections of the traces over the local transition systems satisfy the local properties

Lemma 8 *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be n compatible ITS with function γ_S defined according to section 4.1; $\varphi_1, \ldots, \varphi_n$ be local properties of respectively $\mathcal{M}_1, \ldots \mathcal{M}_n$ and $\pi \in \Pi(V)$ be a trace over the symbols of $\mathcal{M} = \gamma_S(\mathcal{M}_1, \ldots, \mathcal{M}_n)$:*

$$Pr_{\mathcal{M}_1}(\pi) \models \varphi_1 \wedge \cdots \wedge Pr_{\mathcal{M}_n}(\pi) \models \varphi_n \wedge \pi \models \Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n) \Leftrightarrow \pi \in \mathscr{L}(\gamma_S(\mathcal{M}_{\varphi_1}, \ldots, \mathcal{M}_{\varphi_n}))_{|V}$$

where $\mathcal{M}_{\varphi_1}, \ldots, \mathcal{M}_{\varphi_n}$ are respectively the ITS generated applying $LTL2IntTS$ to the symbols of $\mathcal{M}_1, \ldots, \mathcal{M}_n$ and the properties $\varphi_1, \ldots, \varphi_n$.

From lemma 8 we derive the following theorem which states that this approach is equivalent with the one based on rewriting.

Theorem 3. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be n compatible ITS, $\varphi_1, \ldots, \varphi_n$ be local properties of respectively $\mathcal{M}_1, \ldots \mathcal{M}_n$ and $\pi \in \Pi(V)$ be a trace over the symbols of $\mathcal{M} = \gamma_S(\mathcal{M}_1, \ldots, \mathcal{M}_n)$:*

$$\pi \models \gamma_P(\varphi_1, \ldots, \varphi_n) \Leftrightarrow \pi \in \mathscr{L}(\gamma_S(\mathcal{M}_{\varphi_1}, \ldots, \mathcal{M}_{\varphi_n}))_{|V}$$

where $\mathcal{M}_{\varphi_1}, \ldots \mathcal{M}_{\varphi_n}$ are respectively the ITS generated applying $LTL2IntTS$ to the symbols of $\mathcal{M}_1, \ldots, \mathcal{M}_n$ and the properties $\varphi_1, \ldots, \varphi_n$.

6 Experimental evaluation [2]

The techniques of this paper are implemented inside the contract based design tool OCRA [7] and have been validated through an empirical verification of the rewriting theorems. We implemented a technique that applies Pr and Pr^{-1} to lazo-shaped traces generated from LTL formulas to verify the theorems of the rewritings. Moreover, we also checked that the alternative approach was equivalent to the one proposed for LTL. The validation have been conducted on all LTL specifications of the discrete time example models of OCRA (~ 300 formulas) and on 100 randomly generated formulas.[3] We also confronted the approaches with an experimental evaluation.

For completeness, the experimental evaluation considers another technique based on the rewriting of [4] that was already implemented in OCRA and mentioned in section 4.2. We call this rewriting output-only rewriting. Output-only rewriting considers only specifications with local variables and synchronization events. While, to keep the notation readable we did not mention events inside our rewriting, we handle events in our implementation similarly to next operators. To force synchronisation between events, we augment Ψ_{cond} to enable shared events only when its components do not stutter. Due to the limitations of output-only rewriting, the experimental evaluation have been applied only to a sub-set of models. The experiments were run in parallel on a cluster with nodes with Intel Xeon CPU running at 2.27GHz with 8CPU, 48GB. The timeout for each run was four hours and the memory cap was set to 1GB.

The evaluation was applied on different type of models: asynchronous versions of OCRA models, Dwyer LTL patterns [14] parametrized on the number of components and on components with parametrized nested X formulas. The one based on Dwyer LTL patterns [14] considered 3 LTL patterns: *response*, *precedence chain* and *universality* patterns. The models compose the pattern formulas in two ways: as a sequence of n components linked in a bus and as a set of components that tries to write on the output port concurrently. Since the output-only rewriting does not support input port, in the models used in the comparison with the output-only rewriting replace input data readings with synchronizing event exchanging such data.

Each model have been tested with two symbolic model checking algorithms: ic3ia [8] and one based on bdd [23] (only for finite state models) that we will call bdd for brevity; however, due to the limited space we show only plots with the ic3ia algorithm. Figure 1 shows the results of response pattern model with events, universality sequence pattern model with input port and precedence chain model where each component concurrently writes to the global output port.

The experimental evaluation based on nested X sequence considered 2 parameters: the number of nested X of the global property and the number of

[2] The tar files of the experimental evaluation results can be found at: https://es-static.fbk.eu/people/bombardelli/papers/nfm22/expeval.tar.gz

[3] The detailed algorithms of the validation can be found in the appendix of the extended version of this paper at:
https://es-static.fbk.eu/people/bombardelli/papers/nfm22/nfm-extended.pdf

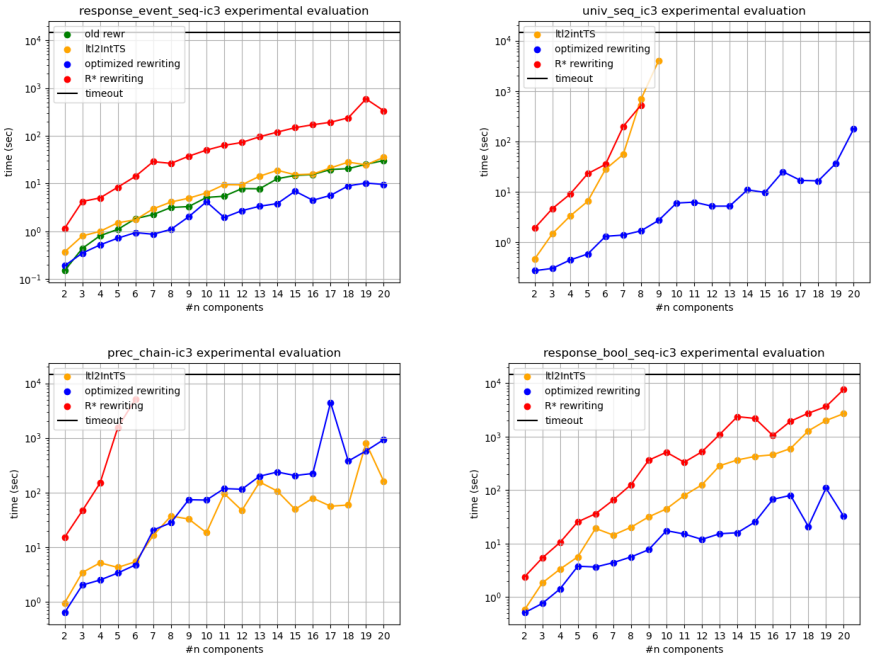
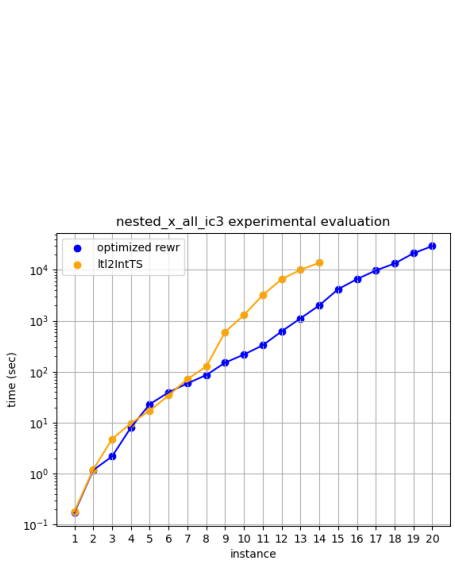
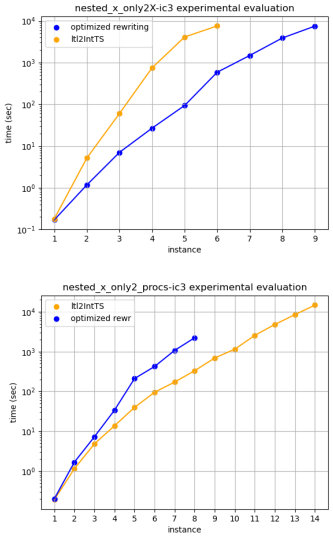


Fig. 1: Pattern experimental evaluations



(a) Overall incremental results



(b) Results with respectively fixed X and fixed components

Fig. 2: Nested X experimental evaluation

components. In this scenario, we confront the approach based on asynchronous composition of ITS with the optimized temporal rewriting. The global property is defined as: $G(G^{\leq n*s}t \rightarrow F^{\leq n*s}r)$ where n is the number of nested X, s is the number of components in the system and t and r are two boolean formulas. Local properties are defined as $G(G^{\leq n}t \rightarrow F^{\leq n}r)$ where n. Figure 2a shows the overall results of the experimental evaluation, where the y-axis represents the time required to check x global properties while figure 2b shows the result restricted to models with $n = 2$ and with $s = 2$.

Figure 3 shows the overall results with scatter plots that confronts the optimized rewriting with the other approaches. In these plots, the y coordinate represents the time to verify the validity of each instance with the *optimized* rewriting while the x coordinate represents the time to verify the validity of each instance with the adversarial approach. If a point is above the dashed line, then the adversarial method performed better; otherwise, the optimized rewriting was faster in verifying the validity of that instance. The optimized rewriting ($\mathcal{R}^{\theta*}$) outperforms the non optimized one ($\mathcal{R}^{*}$) in almost every model. Intuitively, $\mathcal{R}^{\theta*}$ generates formulas that are smaller than those produced by $\mathcal{R}^{*}$ (see example 3 for a comparison between the rewritten formulas). When dealing with nested X, the approach based on asynchronous composition performs better than the optimized rewriting when there are only two components; this is outlined in figure 2b. However, even if $ltl2IntTS$ sometimes performs better, in general the optimized rewriting is faster and is able to solve more instances. The comparison between optimized rewriting and output-only rewriting shows that in general the optimized rewriting is faster. This happens because $\mathcal{R}^{\theta*}$ exploits the absence of input data port to minimize the rewritten formula. Thus, compared with the output-only, $\mathcal{R}^{\theta*}$ is both more general and efficient. To summarize: the optimization significantly improves the performance of the rewriting, the optimization is in general faster than the output-only rewriting, and, apart from certain cases, the optimized rewriting is faster than the approach based on the compilation into interface transition systems.

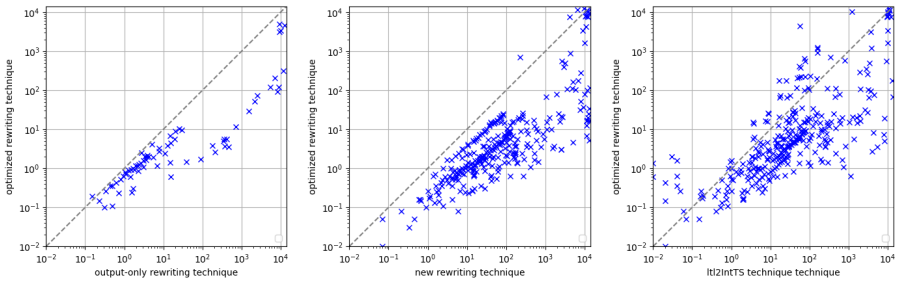


Fig. 3: Scatter plots on all the experiments

7 Conclusions

In this paper, we considered the problem of compositional reasoning for asynchronous systems with LTL properties over input and output variables. We proposed a new rewriting of LTL formulas that allows for checking compositional rules with temporal satisfiability solvers. We provided an optimized version and an alternative solutions based on the compilation of the LTL formulas into transition systems. We finally compare these rewritings con various benchmarks showing the scalability of the approach.

In the future, we will consider various directions for extending the framework including real-time and hybrid specifications, optimizations based on the scheduling of components and other communication mechanisms such as buffered communication, and the application of the proposed rewriting in an extension of Asynchronous HyperLTL [3].

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
2. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, Jan. 2009.
3. J. Baumeister, N. Coenen, B. Bonakdarpour, B. Finkbeiner, and C. Sánchez. A Temporal Logic for Asynchronous Hyperproperties. In *CAV (1)*, volume 12759 of *Lecture Notes in Computer Science*, pages 694–717. Springer, 2021.
4. N. Benes, L. Brim, I. Cerná, J. Sochor, P. Vareková, and B. Buhnova. Partial order reduction for state/event ltl. In *IFM*, 2009.
5. L. Bu, A. Cimatti, X. Li, S. Mover, and S. Tonetta. Model checking of hybrid systems using shallow synchronization. In J. Hatcliff and E. Zucca, editors, *Formal Techniques for Distributed Systems*, pages 155–169, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
6. R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuxmv symbolic model checker. volume 8559, pages 334–342, 07 2014.
7. A. Cimatti, M. Dorigatti, and S. Tonetta. Ocra: A tool for checking the refinement of temporal contracts. pages 702–705, 11 2013.
8. A. Cimatti and A. Griggio. Software model checking via ic3. In *CAV*, pages 277–293, 2012.
9. A. Cimatti, A. Griggio, E. Magnago, M. Roveri, and S. Tonetta. Smt-based satisfiability of first-order ltl with event freezing functions and metric operators. *Information and Computation*, 272:104502, 12 2019.
10. A. Cimatti, S. Mover, and S. Tonetta. Hydi: A language for symbolic hybrid systems with discrete interaction. pages 275–278, 08 2011.
11. A. Cimatti and S. Tonetta. Contracts-refinement proof system for component-based embedded systems. *Science of Computer Programming*, 97:333–348, 2015. Object-Oriented Programming and Systems (OOPS 2010) Modeling and Analysis of Compositional Software (papers from EUROMICRO SEAA 12).
12. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at ltl model checking. Technical report, USA, 1994.

13. L. de Alfaro and T. A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120. ACM, 2001.
14. M. Dwyer, G. Avrunin, and J. Corbett. Patterns in property specifications for finite-state verification. *Proceedings - International Conference on Software Engineering*, 02 1970.
15. C. Eisner, D. Fisman, J. Havlicek, A. McIsaac, and D. V. Campenhout. The Definition of a Temporal Clock Operator. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 857–870. Springer, 2003.
16. B. Jonsson and Y.-K. Tsay. Assumption/guarantee specifications in linear-time temporal logic. *Theor. Comput. Sci.*, 167:47–72, 1996.
17. L. Lamport. Temporal logic of actions. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16:872–923, 05 1994.
18. L. Lamport. The operators of tla. 06 1997.
19. O. Lichtenstein, A. Pnueli, and L. Zuck. The Glory of the Past. In *Logics of Programs*, pages 196–218, 1985.
20. Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992.
21. Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992.
22. K. L. McMillan. Circular Compositional Reasoning about Liveness. In *CHARME*, volume 1703 of *Lecture Notes in Computer Science*, pages 342–345. Springer, 1999.
23. C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Applications*. 01 1998.
24. C. S. Pasareanu, M. B. Dwyer, and M. Huth. Assume-Guarantee Model Checking of Software: A Comparative Case Study. In *SPIN*, volume 1680 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 1999.
25. A. Pnueli. The temporal logic of programs. pages 46–57, 09 1977.
26. W.-P. Roever, F. Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, and J. Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. 01 2001.
27. K. Y. Rozier. Linear temporal logic symbolic model checking. *Computer Science Review*, 5(2):163–203, 2011.
28. O. Rysavy and J. Rab. A formal model of composing components: The tla+ approach. *Innovations in Systems and Software Engineering*, 5:139–148, 06 2009.
29. S. Tonetta. Linear-time Temporal Logic with Event Freezing Functions. In *GandALF*, volume 256 of *EPTCS*, pages 195–209, 2017.

A Rewriting proofs

This section reports the proofs of lemmas and theorems and is appended only as additional information for the reviewers.

A.1 Proof of lemma 1

Proof. The proof of lemma 1 is by induction on the length of the formula φ.

Base case: $\varphi = a$, where $a \in V$:

1. Case a is a *term* or a *constant*:
   - By $\mathcal{R}$ definition $\mathcal{R}(a) = a$
   - By *map* and Pr^{-1} definition $\pi^{ST}(j)(a) = \pi(i)(a)$
2. Case a is a 0-arity *predicate*
   - By $\mathcal{R}$ definition $\mathcal{R}(a) = a$
   - By *map* and Pr^{-1} definition $\pi^{ST}(j)(a) = \pi(i)(a)$ which implies that $\pi^{ST}, j \models a \Leftrightarrow \pi, i \models a$

The previous steps suffice in demonstrating the base case. The following paragraphs demonstrates the inductive case. In all the cases, we exploit the fact that:

- *map* is defined over $\mathbb{N}$ for the traces in $Pr^{-1}(\pi)$ where π is an infinite trace
- *map* is monotonic

We will refer to map^{-1} as the inverse function of *map*. Since *map* is not *surjective*, map^{-1} might not be defined in some points in $\mathbb{N}$. However, during our demonstrations we will use $map^{-1}(j)$ only on $j \in map$, and thus, the existence of $map^{-1}(j)$ in these cases is always granted.

Case $\varphi := \neg\psi$

1. By $\mathcal{R}$ definition: $\mathcal{R}(\neg\psi) = \neg\mathcal{R}(\psi)$
2. By $\neg$ definition $\pi^{ST}, j \models \neg\mathcal{R}(\psi) \Leftrightarrow \pi^{ST}, j, \not\models \mathcal{R}(\psi)$
3. By induction hypothesis $\pi^{ST}, j \not\models \mathcal{R}(\psi) \Leftrightarrow \pi, i \not\models \psi \Leftrightarrow \pi, i \models \neg\psi$

Case $\varphi := \psi \vee \psi'$

1. By $\mathcal{R}$ definition: $\mathcal{R}(\psi \vee \psi') = \mathcal{R}(\psi) \vee \mathcal{R}(\psi')$
2. By $\vee$ definition $\pi^{ST}, j \models \mathcal{R}(\psi) \vee \mathcal{R}(\psi') \Leftrightarrow \pi^{ST}, j \models \mathcal{R}(\psi)$ or $\pi^{ST}, j \models \mathcal{R}(\psi')$
3. By induction hypothesis $\pi^{ST}, j \models \mathcal{R}(\psi)$ or $\pi^{ST}, j \models \mathcal{R}(\psi') \Leftrightarrow \pi, i \models \psi$ or $\pi, i \models \psi' \Leftrightarrow \pi, i \models \psi \vee \psi'$

Case $\varphi := X\psi$

1. By $\mathcal{R}$ definition: $\mathcal{R}(X\psi) = X(\neg stR(st \vee \psi))$
2. By X definition: $\pi^{ST}, j \models X(\neg stR(st \vee \mathcal{R}(\psi))) \Leftrightarrow \pi^{ST}, j+1 \models \neg stR(st \vee \mathcal{R}(\psi))$
3. By R definition: $\pi^{ST}, j+1 \models \neg stR(st \vee \psi) \Leftrightarrow \exists k \geq j+1 : \pi^{ST}, k \models \neg st$ and $\forall j+1 \leq j' \leq k : \pi^{ST}, j' \models st$ or $\pi^{ST}, j' \models \mathcal{R}(\psi)$
4. By π^{ST} definition for all $l \in \mathbb{N} : \pi^{ST}, l \models st \Leftrightarrow l \notin map_{\pi^{ST}}$ hence: $\pi^{ST}, j \models \mathcal{R}(X\psi) \Leftrightarrow \exists k \geq j+1. k \in map_{\pi^{ST}}$ and $\pi^{ST}, k \models \mathcal{R}(\psi)$ and $\forall j+1 \leq j' < k. k \notin map_{\pi^{ST}} \Leftrightarrow \pi^{ST}, map(i+1) \models \mathcal{R}(\psi)$

Case $\varphi := \psi U \psi'$*:*

1. By $\mathcal{R}$ definition: $\mathcal{R}(\psi U \psi') = (st \vee \mathcal{R}(\psi)) U (\neg st \wedge \mathcal{R}(\psi'))$
2. By U definition $\pi^{ST}, j \models (st \vee \mathcal{R}(\psi)) U (\neg st \wedge \mathcal{R}(\psi')) \Leftrightarrow$
   $\exists k \geq j : \pi^{ST}, k \models \neg st \wedge \mathcal{R}(\psi')$ and
   $\forall j \leq l < k : \pi^{ST}, l \models st \vee \mathcal{R}(\psi)$
3. Since for all $j' \pi^{ST}, j' \models st \Leftrightarrow k \notin map_{\pi^{ST}}$ it follows that:
   $\pi^{ST}, j \models \mathcal{R}(\psi U \psi') \Leftrightarrow$
   $\exists k \geq j : k \in map_{\pi^{ST}}$ and $\pi^{ST}, k \models \mathcal{R}(\psi')$ and
   $\forall j \leq l < k : l \in map_{\pi^{ST}}$ implies $\pi^{ST}, l \models \mathcal{R}(\psi)$
4. By induction hyphotesis it follows that: $\pi^{ST}, j \models \mathcal{R}(\psi U \psi') \Leftrightarrow$
   $\exists k \geq j : k \in map_{\pi^{ST}}$ and $\pi, map^{-1}_{\pi^{ST}}(k) \models \psi'$ and
   $\forall j \leq l < k : l \in map_{\pi^{ST}}$ implies $\pi, map^{-1}_{\pi^{ST}}(l) \models \psi \Leftrightarrow$
   $\exists k' \geq i : \pi, k' \models \psi'$ and $\forall i \leq l' < k' : \pi, l' \models \psi \Leftrightarrow \pi, i \models \psi U \psi'$

Case $\varphi := Y\psi$*:*

1. By $\mathcal{R}$ definition: $\mathcal{R}(Y\psi) = Y(stS(\neg st \wedge \mathcal{R}(\psi)))$
2. By Y definition:
   $\pi^{ST}, j \models \mathcal{R}(\psi) \Leftrightarrow \pi^{ST}, j-1 \models stS(\neg st \wedge \mathcal{R}(\psi))$ and $j > 0$
3. If $j = 0$ then $i = 0$ hence both are not satisfied.
4. Otherwise by S definition: $\pi^{ST}, j-1 \models stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow$
   $\exists k \leq j-1 : \pi^{ST}, k \models \neg st \wedge \mathcal{R}(\psi)$ and $\forall k < j' \leq j-1 : \pi^{ST}, j' \models st$
5. Since for all $k \in \mathbb{N} : \pi^{ST}, k \models st \Leftrightarrow k \notin map_{\pi^{ST}}$ it follows that: $\pi^{ST}, j-1 \models stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow$
   $\exists k \leq j-1 : k \in map_{\pi^{ST}}$ and $\pi^{ST}, k \models \mathcal{R}(\psi)$ and
   $\forall k < j' \leq j-1 : j' \notin map_{\pi^{ST}}$
6. Here there are two possibilities:
   (a) $i = 0$: $\exists j' \leq j-1 : j' \in map_{\pi^{ST}}$ does not hold as expected.
   (b) $i > 0$: Then i has a previous state, so
   $\pi^{ST}, j-1 \models stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow$
   $\pi^{ST}, k \models \mathcal{R}(\psi)$ where $k = map_{\pi^{ST}}(i-1)$

Case $\varphi := \psi S \psi'$*:*

1. By $\mathcal{R}$ definition:
   $\mathcal{R}(\psi S \psi') = (\psi \vee st) S (\neg st \wedge \psi)$
2. By S definition it follows that:
   $\pi^{ST}, j \models (st \vee \mathcal{R}(\psi) S (\neg st \wedge \mathcal{R}(\psi'))) \Leftrightarrow$
   $\exists k \leq j : \pi^{ST}, k \models (\neg st \wedge \psi')$ and $\forall k < l \leq j : \pi^{ST}, l \models st \vee \psi$
3. Since for all $k \in \mathbb{N} : \pi^{ST} \models st \Leftrightarrow k \notin map_{\pi^{ST}}$ it follows that:
   $\pi^{ST}, j \models \mathcal{R}(\psi S \psi') \Leftrightarrow$
   $\exists k \leq j : k \in map_{\pi^{ST}}$ and $\pi^{ST}, k \models \mathcal{R}(\psi')$ and
   $\forall k < l \leq j : l \notin map_{\pi^{ST}}$ or $\pi^{ST}, l \models \mathcal{R}(\psi) \Leftrightarrow$
   $\exists k' \leq i : \pi^{ST}, map_{\pi^{ST}}(k') \models \mathcal{R}(\psi')$ and
   $\forall k' < l' \leq i : \pi^{ST}, map_{\pi^{ST}}(l') \models \mathcal{R}(\psi) \Leftrightarrow$
   $\exists k' \leq i : \pi, k' \models \psi'$ and $\forall k' < l' \leq i : \pi, l' \models \psi$

Case $\varphi := ite(\psi, \psi_1, \psi_2)$

1. By $\mathcal{R}$ definition $\mathcal{R}(ite(\psi, \psi_1, \psi_2)) = ite(\mathcal{R}(\psi), \mathcal{R}(\psi_1), \mathcal{R}(\psi_2))$
2. By *ite* definition $\pi^{ST}(j)(ite(\mathcal{R}(\psi), \mathcal{R}(\psi_1), \mathcal{R}(\psi_2)))) = \pi^{ST}(j)(\mathcal{R}(\psi_1))$ if $\pi^{ST}, j \models \mathcal{R}(\psi), \mathcal{R}(\psi_2)$ otherwise
3. By induction hypothesis $\pi^{ST}, j \models \mathcal{R}(\psi) \Leftrightarrow \pi, i \models \psi$ and $\pi^{ST}(j)(\mathcal{R}(\pi_1^{ST})) = \pi(i)(\pi_1^{ST})$ and $\pi^{ST}(j)(\mathcal{R}(\pi_2^{ST})) = \pi(j)(\pi_2^{ST})$

Case $\varphi := next(\psi)$

1. By $\mathcal{R}$ definition $\mathcal{R}(next(\psi)) = \mathcal{R}(\psi) @ \tilde{F} \neg st$
2. By $@\tilde{F}$ definition $\pi^{ST}(j)(\mathcal{R}(\psi) @ \tilde{F} \neg st) = \pi^{ST}(j')(\mathcal{R}(\psi))$ if there exists $j' > j : \forall j < j'' < j' : \pi^{ST}, j'' \not\models \neg st$ and $\pi^{ST}, j' \models \neg st$
3. By Pr^{-1} and *map* definitions j'' exists, since *map* is *stricly monotonic* it follows that: $j'' = map_{\pi^{ST}}(i{+}1)$ hence: $\pi^{ST}(j)(\mathcal{R}(next(\psi))) = \pi^{ST}(map_{\pi^{ST}}(i{+}1))(\mathcal{R}(\psi)) = \pi(i+1)(\psi) = \pi(i)(next(\psi))$

Case $\varphi := \psi_1 @ \tilde{F} \psi_2$

1. By $\mathcal{R}$ definition $\mathcal{R}(\psi_1 @ \tilde{F} \psi_2)) = \mathcal{R}(\psi_1) @ \tilde{F}(\psi_2 \wedge \neg st)$
2. By $@\tilde{F}$ definition $\pi^{ST}(j)(\mathcal{R}(\psi_1) @ \tilde{F}(\neg st \wedge \mathcal{R}(\psi_2)) = \pi^{ST}(j')(\mathcal{R}(\psi_1))$ if there exists $j' > j : \forall j < j'' < j' : \pi^{ST}, j'' \not\models \neg st \wedge \mathcal{R}(\psi_2)$ and $\pi^{ST}, j' \models \neg st \wedge \mathcal{R}(\psi_2)$
3. By Pr^{-1} and *map* definitions if j'' exists $j'' \in map_{\pi^{ST}}$ hence: $\pi^{ST}(j)(\mathcal{R}(\psi_1 @ \tilde{F} \psi_2)) = \pi^{ST}(j')(\mathcal{R}(\psi_1))$ if there exists $j' > j : \pi^{ST}, j' \models \mathcal{R}(\psi_1)$ and $j' \in map_{\pi^{ST}}$ and for all $j < j'' < j' : j' \notin map_{\pi^{ST}}$ or $\pi^{ST}, j'' \not\models \mathcal{R}(\psi_2)$
4. By induction hypothesis and since *map* is *strictly monotonic* if j'' exists: $j'' = map_{\pi^{ST}}(i'')$ and $j' = map_{\pi^{ST}}(i')$ and $i < i' < i''$, hence: $\pi^{ST}(j)(\mathcal{R}(\psi_1 @ \tilde{F} \psi_2)) = \pi(i)(\psi_1 @ \tilde{F} \psi_2)$

Case $\varphi := \psi_1 @ \tilde{P} \psi_2$

1. By $\mathcal{R}$ definition $\mathcal{R}(\psi_1 @ \tilde{P} \psi_2)) = \mathcal{R}(\psi_1) @ \tilde{P}(\psi_2 \wedge \neg st)$
2. By $@\tilde{P}$ definition $\pi^{ST}(j)(\mathcal{R}(\psi_1) @ \tilde{P}(\neg st \wedge \mathcal{R}(\psi_2)) = \pi^{ST}(j')(\mathcal{R}(\psi_1))$ if there exists $j' < j : \forall j > j'' > j' : \pi^{ST}, j'' \nvDash \neg st \wedge \mathcal{R}(\psi_2)$ and $\pi^{ST}, j' \models \neg st \wedge \mathcal{R}(\psi_2)$
3. By Pr^{-1} and *map* definitions if j'' exists $j'' \in map_{\pi^{ST}}$ hence: $\pi^{ST}(j)(\mathcal{R}(\psi_1 @ \tilde{P} \psi_2)) = \pi^{ST}(j')(\mathcal{R}(\psi_1))$ if there exists $j' < j : \pi^{ST}, j' \models \mathcal{R}(\psi_2)$ and $j' \in map_{\pi^{ST}}$ and for all $j > j'' > j' : j'' \notin map_{\pi^{ST}}$ or $\pi^{ST}, j'' \nvDash \mathcal{R}(\psi_2)$
4. By induction hypothesis and since *map* is *strictly monotonic* if j'' exists: $j'' = map_{\pi^{ST}}(i'')$ and $j' = map_{\pi^{ST}}(i')$ and $i > i' > i''$, hence: $\pi^{ST}(j)(\mathcal{R}(\psi_1 @ \tilde{P} \psi_2)) = \pi(i)(\psi_1 @ \tilde{P} \psi_2)$

A.2 Proof of lemma 2

Proof. 1. $\pi^{ST}, 0 \models \neg st R(st \vee \mathcal{R}(\varphi)) \Leftrightarrow \exists l \geq 0, \forall 0 \leq k \leq l : \pi^{ST}, k \models st \vee \mathcal{R}(\psi)$ and $\pi^{ST}, l \models \neg st$
2. Since $\forall n : \pi^{ST}, n \models st \Leftrightarrow map_{\pi^{ST}} : \pi^{ST}, 0 \models \neg st R(st \vee \mathcal{R}(\varphi)) \Leftrightarrow \exists l \geq 0, \forall 0 \leq k \leq l : l \in map_{\pi^{ST}}$ and $k \in map_{\pi^{ST}} \rightarrow \mathcal{R}(\varphi)$
3. By *map* definition $l = map_{\pi^{ST}}(0)$ (*map* is defined in $\mathbb{N}$ and is **strictly monotonic**) hence $\pi^{ST}, 0 \models \neg st R(st \vee \mathcal{R}(\varphi)) \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(0) \models \mathcal{R}(\varphi)$

A.3 Proof of theorem 1

Proof. We prove the correctness using lemma 1 and 2:

1. By Lemma 1: $\pi \models \varphi \Leftrightarrow \pi, 0 \models \varphi \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(0) \models \mathcal{R}(\varphi)$
2. By Lemma 2: $\pi^{ST}, map_{\pi^{ST}}(0) \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST} \models \mathcal{R}^*(\varphi)$

A.4 Proof of lemmas on stutter tolerant formulas

Proof for until formulas

Proof. This lemma can be proven by induction on the index j.

- **Base case:** $j = map_{\pi^{ST}}(i)$
  The base case trivially holds since $\pi^{ST}(j) = \pi^{ST}(map_{\pi^{ST}}(i))$
- **Inductive case:** $m_{i-1} = map_{\pi^{ST}}(i-1) < j < m_i = map_{\pi^{ST}}(i)$

  - By $\mathcal{R}$ definition $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST}, j \models (st \vee \mathcal{R}(\psi_1)) U (\neg st \wedge \mathcal{R}(\psi_2))$
  - By U definition $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \exists k \geq j : \pi^{ST}, k \models \neg st \wedge \mathcal{R}(\psi_2)$ and $\forall j \leq l < k : \pi^{ST}, l \models st \vee \mathcal{R}(\psi_1)$
  - Since $j \notin map_{\pi^{ST}}$ then $\pi^{ST}, j \models st$, which implies that $\pi^{ST}, j \models st \vee \mathcal{R}(\psi_1)$ and $\pi^{ST}, j \nvDash \neg st \wedge \mathcal{R}(\psi_2)$ and hence $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST}, j + 1, \models \mathcal{R}(\varphi)$
  - By induction hypothesis then $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \pi^{ST}, m_i \models \mathcal{R}(\varphi)$

Proof for yesterday formulas

Proof. This lemma can be proven by induction on the index j.

- **Base case:** $j = map_{\pi^{ST}}(i)$
  The base case trivially holds since $\pi^{ST}(j) = \pi^{ST}(map_{\pi^{ST}}(i))$
- **Inductive case:** $m_{i-1} = map_{\pi^{ST}}(i-1) < j < m_i = map_{\pi^{ST}}(i)$

  - By induction hypothesis $\pi^{ST}, m_i \models Y(stS(\neg st \wedge \mathcal{R}(\psi))) \Leftrightarrow \pi^{ST}, j+1 \models Y(stS(\neg st \wedge \mathcal{R}(\psi))) \Leftrightarrow \pi^{ST}, j \models stS(\neg st \wedge \mathcal{R}(\psi))$
  - By S definition $\pi^{ST}, stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow \exists k \leq j : \pi^{ST}, k \models \neg st \wedge \mathcal{R}(\psi)$ and $\forall k < l \leq j : \pi^{ST}, l \models st$
  - Since $j \notin map_{\pi^{ST}}$ then $\pi^{ST}, j \models st$ and $\pi^{ST}, j \not\models \neg st$ hence $\pi^{ST}, j \models stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow j > 0$ and $\pi^{ST}, j-1 \models stS(\neg st \wedge \mathcal{R}(\psi)) \Leftrightarrow \pi^{ST}, j \models Y(stS(\neg st \wedge \mathcal{R}(\psi))) \Leftrightarrow \pi^{ST}, j \models \mathcal{R}(\psi)$

Proof for at last formulas

Proof. This lemma can be proven by induction on the index j.

- **Base case:** $j = map_{\pi^{ST}}(i)$
  The base case trivially holds since $\pi^{ST}(j) = \pi^{ST}(map_{\pi^{ST}}(i))$
- **Inductive case:** $m_{i-1} = map_{\pi^{ST}}(i-1) < j < m_i = map_{\pi^{ST}}(i)$

  - By $\mathcal{R}$ definition $\mathcal{R}(\varphi) = \mathcal{R}(\psi_v)@\tilde{P}(\mathcal{R}(\psi_t) \wedge \neg st)$
  - By $@\tilde{P}$ definition $\pi^{ST}(j)(\mathcal{R}(\varphi)) = \pi^{ST}(j')(\mathcal{R}(\psi_v))$ if there exists $j' < j : \pi^{ST}, j' \models \mathcal{R}(\psi_t) \wedge \neg st$ and for all $j > j'' > j' : \pi^{ST}, j'' \not\models \neg st \wedge \mathcal{R}(\psi_t)$
  - Since $j \notin map_{\pi^{ST}}$ it follows that $\pi^{ST}, j \not\models \neg st \wedge \mathcal{R}(\psi_t)$, hence $\pi^{ST}(j+1)(\mathcal{R}(\varphi)) = \pi^{ST}(j)(\mathcal{R}(\varphi))$
  - By induction hypothesis then $\pi^{ST}(j)(\mathcal{R}(\varphi)) = \pi^{ST}(m_i)(\mathcal{R}(\varphi))$

Proof of lemma 4

Proof. We can proof this lemma by induction. Base cases:

- If φ is an *until formula*, by lemma 3 it is stutter tolerant w.r.t. $\mathcal{R}$
- If φ is an *yesterday formula*, by lemma 3 it is stutter tolerant w.r.t. $\mathcal{R}$
- If φ is an *at last formula*,by lemma 3 it is stutter tolerant w.r.t. $\mathcal{R}$
- If $\varphi = s$ and $s \in V_{O \cup H}$ then by Pr^{-1} definition s does not change during stuttering transitions and hence it is *stutter tolerant*

Inductive step:

- $\varphi = \psi_1 \vee \psi_2$:
  1. $\pi^{ST}, j \models \mathcal{R}(\psi_1) \vee \mathcal{R}(\psi_2) \Leftrightarrow \pi^{ST}, j \models \mathcal{R}(\psi_1)$ or $\pi^{ST}, j \models \mathcal{R}(\psi_2)$.
  2. Since by induction hypothesis both $\mathcal{R}(\psi_1)$ and $\mathcal{R}(\psi_2)$ are satisfied in j iff they are satisfied in m_i then also their disjunction is satisfied in j iff it is satisfied in m_i.
- $\varphi = \neg\psi$: This case is trivial: since $\mathcal{R}(\psi)$ is satisfied in j iff it is satisfied in m_i then also its negation will have the same property

A.5 Proof of lemma 5

Proof. As for lemma 1, this lemma is proved by induction on the length of the formula.

- **Base case:** The base case is identical to the one of the proof of lemma 1.
- **Inductive case:** If the sub-formulas are not stutter-tolerant the proof is identical to the one of lemma 1.
  Otherwise, it is necessary to consider the cases in which φ is a *temporal* formula (X, U, S)or *temporal* term $(next, @F)$ composed by *stutter-tolerant* sub-formulas
  1. **Case** $\varphi = X(\psi)$:
     (a) By $\mathcal{R}^\theta$ definition $\mathcal{R}^\theta(\varphi) = X\mathcal{R}^\theta(\psi)$
     (b) By X definition $\pi^{ST}, j \models X\mathcal{R}^\theta(\psi) \Leftrightarrow \pi^{ST}, j+1 \models \mathcal{R}^\theta(\psi)$.
     (c) By map definition there are two possible cases:
        i. $j+1 = map_{\pi^{ST}}(i+1)$: Skip to the next point
        ii. $j+1 \notin map_{\pi^{ST}}$ and $j+1 < map_{\pi^{ST}}(i+1)$: Since ψ is a *stutter-tolerant* formula, $\pi^{ST}, j+1 \models \mathcal{R}^\theta(\psi) \Leftrightarrow \pi^{ST}, map_{\pi^{ST}}(i+1) \models \mathcal{R}^\theta(\psi)$
     (d) By induction hypothesis $\pi^{ST}, map_{\pi^{ST}}(i+1) \models \mathcal{R}^\theta(\psi) \Leftrightarrow \pi, i+1 \models \psi \Leftrightarrow \pi, i \models X\psi$
  2. **Case** $\varphi = \psi_1 U \psi_2$:
     (a) By $\mathcal{R}^\theta$ definition $\mathcal{R}^\theta(\varphi) = \mathcal{R}^\theta(\psi_1)U\mathcal{R}^\theta(\psi_2)$
     (b) By U definition $\pi^{ST}, j \models \mathcal{R}^\theta(\psi_1)U\mathcal{R}^\theta(\psi_2) \Leftrightarrow \exists k \geq j : \pi^{ST}, k \models \mathcal{R}^\theta\psi_2$ and $\forall j \leq l < k : \pi^{ST}, l \models \mathcal{R}^\theta\psi_1$
     (c) Since ψ_1 and ψ_2 are *stutter-tolerant* and since *map* is monotonic it follows that: $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \exists \bar{k} \in map_{\pi^{ST}} \geq j : \pi^{ST}, \bar{k} \models \mathcal{R}^\theta(\psi_1)$ and $\forall j \leq \bar{l} < \bar{k}$ such that $\bar{l} \in map_{\pi^{ST}} : \pi^{ST}, \bar{l} \models \mathcal{R}^\theta(\psi_2)$
     (d) By induction hypothesis $\pi^{ST}, j \models \mathcal{R}^\theta(\varphi) \Leftrightarrow \exists k' \geq i$ where $\bar{k} = map_{\pi^{ST}}(k') : \pi, k \models \psi_1$ and $\forall i \leq l' < k'$ where $\bar{l} = map_{\pi^{ST}}(l') : \pi, l' \models \psi_2 \Leftrightarrow \pi, i \models \psi_1 U \psi_2$
  3. **Case** $\varphi = \psi_1 S \psi_2$:
     (a) By $\mathcal{R}^\theta$ definition $\mathcal{R}^\theta(\varphi) = \mathcal{R}^\theta(\psi_1)S\mathcal{R}^\theta(\psi_2)$
     (b) By S definition $\pi^{ST}, j \models \mathcal{R}^\theta(\psi_1)S\mathcal{R}^\theta(\psi_2) \Leftrightarrow \exists k \leq j : \pi^{ST}, k \models \mathcal{R}^\theta\psi_2$ and $\forall j \geq l > k : \pi^{ST}, l \models \mathcal{R}^\theta\psi_1$
     (c) Since ψ_1 and ψ_2 are *stutter-tolerant* and since *map* is monotonic it follows that: $\pi^{ST}, j \models \mathcal{R}(\varphi) \Leftrightarrow \exists \bar{k} \in map_{\pi^{ST}} \leq j : \pi^{ST}, \bar{k} \models \mathcal{R}^\theta(\psi_1)$ and $\forall j \geq \bar{l} > \bar{k}$ such that $\bar{l} \in map_{\pi^{ST}} : \pi^{ST}, \bar{l} \models \mathcal{R}^\theta(\psi_2)$
     (d) By induction hypothesis $\pi^{ST}, j \models \mathcal{R}^\theta(\varphi) \Leftrightarrow \exists k' \leq i$ where $\bar{k} = map_{\pi^{ST}}(k') : \pi, k \models \psi_1$ and $\forall i \geq l' > k'$ where $\bar{l} = map_{\pi^{ST}}(l') : \pi, l' \models \psi_2 \Leftrightarrow \pi, i \models \psi_1 S \psi_2$
  4. **Case** $\varphi = next(\psi)$:
     (a) By $\mathcal{R}^\theta$ definition $\mathcal{R}^\theta(\varphi) = next(\mathcal{R}^\theta(\psi))$
     (b) By *next* definition $\pi^{ST}(j)(next(\mathcal{R}^\theta(\psi))) = \pi^{ST}(j+1)(\mathcal{R}^\theta(\psi))$.
     (c) By map definition there are two possible cases:
        i. $j+1 = map_{\pi^{ST}}(i+1)$: Skip to the next point

ii. $j+1 \notin map_{\pi^{ST}}$ and $j+1 < map_{\pi^{ST}}(i+1)$: Since ψ is a *stutter-tolerant* term or formula, $\pi^{ST}(j+1)(\mathcal{R}^{\theta}(\psi)) = \pi^{ST}(map_{\pi^{ST}}(i+1))(\mathcal{R}^{\theta}(\psi))$

(d) By induction hypothesis $\pi^{ST}(map_{\pi^{ST}}(i+1))(\mathcal{R}^{\theta}(\psi)) = \pi(i+1)(\psi) = \pi(i)(next(\psi))$

5. **Case** $\varphi = \psi_1 @ F\psi_2$: In this proof the case in which ψ_2 never hold is not taken into account since it is not relevant.
   (a) By $\mathcal{R}^{\theta}$ definition $\mathcal{R}^{\theta}(\varphi) = \mathcal{R}^{\theta}(\psi_1)@F\mathcal{R}^{\theta}(\psi_2)$
   (b) By $@F$ definition $\pi^{ST}(j)(\mathcal{R}^{\theta}(\psi_1)@F\mathcal{R}^{\theta}(\psi_2)) = \pi^{ST}(j')(\mathcal{R}^{\theta}(\psi_1))$if there exists $j' > j : \pi^{ST}, j' \models \mathcal{R}^{\theta}(\psi_2)$ and $\forall j < j'' < j\pi^{ST}, j'' \not\models \mathcal{R}^{\theta}(\psi_2)$
   (c) Since ψ_1 and ψ_2 are *stutter-tolerant* and since *map* is monotonic it follows that: $\pi^{ST}(j)(\mathcal{R}^{\theta}(\varphi)) = \pi^{ST}(\bar{j'})(\mathcal{R}^{\theta})$if there exists $\bar{j'} > j : \bar{j'} \in map_{\pi^{ST}}$ and $\pi^{ST}, \bar{j'} \models \mathcal{R}^{\theta}(\psi_2)$and $\forall j < \bar{j''} < \bar{j'} : \bar{j''} \in map_{\pi^{ST}}$ and $\pi^{ST}, j'' \not\models \mathcal{R}^{\theta}(\psi_2)$
   (d) By induction hypothesis $\pi^{ST}(j)(\mathcal{R}^{\theta}(\varphi)) = \pi(i')(\psi_1)$ if there exists i' where $\bar{j'} = map_{\pi^{ST}}(i') : \pi, i' \models \psi_2$ and $\forall i < i'' < i'$ where $\bar{j''} = map_{\pi^{ST}}(i'') : \pi, i'' \not\models \psi_2 = \pi(i)(\psi_1 @ F\psi_2)$

A.6 Proof of lemma 6

Proof. The case in which φ is not syntactically stutter-tolerant is identical to lemma 2, while the other case is a special case of the *stutter-tolerant* definition.

A.7 Proof of theorem 2

Proof. The demonstration is the same of theorem 1. The only difference is that instead of lemma 1 and 2, this lemma uses 5 and 6.

A.8 Proof of lemma 7

Proof. ($\mathcal{M}_{\varphi}$ is an ITS) To be an ITS, $\mathcal{M}_{\varphi}$ must have disjointed input, output and internal variables. Input and output variables are taken from $\mathcal{M}$; thus, they are already disjointed. Internal variables are given by V_H in union with new variables generated through $LTL2TS$, and hence, the intersection with input and output variables is empty. Initial conditions do not contain input variables since these variables are replaced with new internal variables. Transition formulas do not contain input next formulas for the same reason.

($\mathcal{M}_{\varphi} \models \varphi$) Since $LTL2TS(\varphi) \models \varphi$ and the guess variables have the same values of the input variables, then $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{F}$ are equisatisfiable, and thus, both transition systems satisfy φ.

A.9 Proof of lemma 8

Proof. By Lemma 7, for each i, $\mathcal{M}_{\varphi_i} \models \varphi_i$. Thus, $Pr_{\mathcal{M}_i}(\pi) \in \mathscr{L}(\mathcal{M}_{\varphi_i})(V^i)$. We need to prove that $Pr_{\mathcal{M}_1}(\pi) \in \mathscr{L}(\mathcal{M}_{\varphi_1}(V^1)) \wedge \cdots \wedge Pr_{\mathcal{M}_n}(\pi) \in \mathscr{L}(\mathcal{M}_{\varphi_n}(V^n)) \wedge$

$\pi \models \Psi \Leftrightarrow \pi \in \mathscr{L}(\gamma_S(\mathcal{M}_{\varphi_1}, \ldots, \mathcal{M}_{\varphi_n}))$. We prove this for every part of the ITS tuple ($\mathcal{I}$, $\mathcal{T}$ and $\mathcal{F}$). (Case $\mathcal{I}$) In case of $\mathcal{I}$, each $\mathcal{I}_{\varphi_i}$ is composed through the conjunction of the other initial conditions. By ψ_{cond} ($\Leftarrow$) and Ψ ($\Rightarrow$), for all φ, if $\pi, j \models st^i$ then $\pi(V^i)(j) = \pi(V^i)(j+1)$. Since, $\mathcal{F}$ contains each $\neg st^i$ ($\Leftarrow$) and since $\Psi \models GF(\neg st^1) \wedge \ldots GF(\neg st^n)$ ($\Rightarrow$), then there exists $k \geq 0$ s.t. $\pi, k \not\models st^i$ and for all $0 \leq l < k$ $\pi, l \models st^i$. Thus, $\pi, 0 \models \mathcal{I}_i \Leftrightarrow \pi, k \models \mathcal{I}_i$ where k is the first non-stuttering transition, and hence, by Pr definition $\Pr_{\mathcal{M}_i}(\pi)(0) \models \mathcal{I}^i \Leftrightarrow \pi, 0 \models \mathcal{I}^i$. Since $\mathcal{I} = \mathcal{I}_1 \wedge \cdots \wedge \mathcal{I}_n$, $\Pr_{\mathcal{M}_1}(\pi), 0 \models \mathcal{I}_0 \wedge \cdots \wedge \Pr_{\mathcal{M}_n}(\pi), 0 \models \mathcal{I}_n \Leftrightarrow \pi, 0 \models \mathcal{I}$. (Case $\mathcal{T}$) In case of $\mathcal{T}$, since the formulas might contain next variables, we need to demonstrate that expressions over dotted variables holds in the next state in which not stutter holds. By $\mathcal{T}$ composition, for all j s.t. $\pi, j \not\models st_i$: $\pi, j \models \mathcal{T}_i$. If $\mathcal{T}_i$ is a formula over V^i, since we consider only variables in the state j, then $\pi, j \models \mathcal{T}_i \Leftrightarrow Pr_{\mathcal{M}_i}(\pi) \models \mathcal{T}_i$. If $\mathcal{T}_i$ is a formula over $V_O^{i\prime} \cup V_H^{i\prime}$, then it must be evaluated in $j+1$. Since, by ψ_{cond}, for all φ, if $\pi, j' \models st^i$ then $\pi(V^i)(j') = \pi(V^i)(j'+1)$. Since, $\mathcal{F}$ contains each $\neg st^i$, then there exists $k \geq j$ s.t. $\pi, k \not\models st^i$ and for all $j \leq l < k : \pi, l \models st^i$. Thus, $\pi(V^i)(j) = \pi(V^j)(k)$ where k is the next state where $\neg st^i$ holds. This, by Pr definition implies that $\pi, j \models \mathcal{T}^i \Leftrightarrow Pr_{\mathcal{M}^i}(\pi) \models \mathcal{T}^i$. The case with formulas with both dotted and non dotted formula can be seen as the inductive case and trivially holds. Since $\mathcal{T}$ is the conjunction of $\neg st^i \rightarrow \mathcal{T}^i$ and their ψ_{cond}, the conjunction trivially holds. (Case $\mathcal{F}$) In case of $\mathcal{F}$, each fairness condition is included in conjunction with not stutter. Since fairness constraints are evaluated as infinitely often φ, then their conjunction with not stutter restricts such constraints to the states in which each $\neg st^i$ holds. Thus, by Pr definition, for all $\varphi_i \in \mathcal{F}_i$, for all j, exists k s.t. $\pi, k \models \neg st \wedge \varphi_i \Leftrightarrow Pr_{\mathcal{M}^i}(\pi), k' \models \varphi_i$.

A.10 Proof of theorem 3

Proof. γ_P is composed of the rewriting in conjunction with Ψ_{cond}:
$\pi \models \gamma_P(\varphi_1, \ldots, \varphi_n) \Leftrightarrow \pi \models \mathcal{R}^*_{\mathcal{M}_1}(\varphi_1) \wedge \cdots \wedge \mathcal{R}^*_{\mathcal{M}_n}(\varphi_n) \wedge \Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n)$. By theorem 1 and since Pr is the inverse of Pr^{-1}, for each $i : \pi \models \mathcal{R}^*_{\mathcal{M}_i}(\varphi_i) \Leftrightarrow Pr_{\mathcal{M}_i}(\pi) \models \varphi_i$. Thus, $\pi \models \gamma_P(\varphi_1, \ldots, \varphi_n) \Leftrightarrow Pr_{\mathcal{M}_1}(\pi) \models \varphi_1 \wedge \cdots \wedge Pr_{\mathcal{M}_n}(\pi) \models \varphi_n \wedge \pi \models \Psi_{cond}(\mathcal{M}_1, \ldots, \mathcal{M}_n)$. By lemma 8 the theorem holds.