Action-Based Model Checking: Logic, Automata, and Reduction

Stephen F. Siegel and Yihao Yan*

University of Delaware, Newark DE 19716, USA
{siegel,yihaoyan}@udel.edu

Abstract. Stutter invariant properties play a special role in state-based model checking: they are the properties that can be checked using partial order reduction (POR), an indispensable optimization. There are algorithms to decide whether an LTL formula or Büchi automaton (BA) specifies a stutter-invariant property, and to convert such a BA to a form that is appropriate for on-the-fly POR-based model checking.
The *interruptible* properties play the same role in action-based model checking that stutter-invariant properties play in the state-based case. These are the properties that are invariant under the insertion or deletion of "invisible" actions. We present algorithms to decide whether an LTL formula or BA specifies an interruptible property, and show how a BA can be transformed to an *interrupt normal form* that can be used in an on-the-fly POR algorithm. We have implemented these algorithms in a new model checker named McRERS, and demonstrate their effectiveness using the RERS 2019 benchmark suite.

1 Introduction

To apply model checking to a concurrent system, one must formulate properties that the system is expected to satisfy. A property may be expressed by specifying acceptable sequences of states, or by specifying acceptable sequences of actions—the events that cause the state to change. Each approach has advantages and disadvantages, and in any particular context one may be more appropriate than the other.

In the state-based context, there is a rich theory involving automata, logic, and reduction for model checking. Some of the core ideas in this theory can be summarized as follows. First, the behavior of the concurrent system is represented by a state-transition system T. One identifies a set AP of atomic propositions, and each state of T is labeled by the set of propositions which hold at that state. An execution passes through an infinite sequence of states, which defines a *trace*, i.e., a sequence of subsets of AP. A *property* is a set of traces, and T satisfies the property if every trace of T is in P.

Properties may be specified by formulas in a temporal logic, such as LTL [21]. There are algorithms (e.g., [31]) to convert an LTL formula ϕ to an equivalent

---
* Currently employed at Google

Büchi automaton (BA) B_ϕ with alphabet 2^{AP}. (Properties may also be specified directly using BAs.) The system T satisfies ϕ if and only if the language of the synchronous product $T \otimes B_{\neg\phi}$ is empty. The emptiness of the language can be determined on-the-fly, i.e., while the reachable states of the product are being constructed.

A property P is *stutter-invariant* if it is closed under the insertion and deletion of repetitions, i.e., $s_0 s_1 \cdots \in P \;\Leftrightarrow\; s_0^{i_0} s_1^{i_1} \cdots \in P$ holds for any positive integers $i_0, i_1, \cdots$. Many algorithms are known for deciding whether an LTL formula or a BA specifies a stutter-invariant property [17,19]. There is also an argument that only stutter-invariant properties should be used in practice. For example, suppose that a trace is formed by sampling the state of a system once every millisecond. If we sample the same system twice each millisecond, and there are no state changes in the sub-millisecond intervals, the second trace will be stutter-equivalent to the first. A meaningful property should be invariant under this choice of time resolution.

Stutter-invariant properties are desirable for another reason: they admit the most significant optimization in model checking, partial order reduction (POR, [10,18,20]). At each state encountered in the exploration of the product space, an on-the-fly POR scheme produces a subset of the enabled transitions. Restricting the search to the transitions in those subsets does not affect the language emptiness question. Recent work has revealed that the BA must have a certain form—"SI normal form"—when POR is used with on-the-fly model checking, but any BA with a stutter-invariant language can be easily transformed into SI normal form [22].

The purpose of this paper is to elaborate an analogous theory for actions. We call the class of properties in the action context that are analogous to the stutter-invariant properties in the state context the *interruptible* properties (Section 3). These properties are invariant under "action stuttering" [28], i.e., the insertion or deletion of "invisible" actions. We present algorithms for deciding whether an LTL formula or a BA specifies an interruptible property (Theorems 1 and 2); to the best of our knowledge, these are the first published algorithms for deciding this property of formulas or automata.

Interruptible properties play the same role in action-based POR that stutter-invariant properties play in state-based POR. In particular, we present an action-based on-the-fly POR algorithm that works for interruptible properties (Section 4). As with the state-based case, the algorithm requires that the BA be in a certain normal form. We introduce a novel *interrupt normal form* (Def. 11) for this purpose, and show how any BA with an interruptible language can be transformed into that form. The relation to earlier work is discussed in Section 5. The effectiveness of these reduction techniques is demonstrated by applying them to problems in the 2019 RERS benchmark suite (Section 6).

2 Preliminaries

Let S be a set. 2^S denotes the set of all subsets of S. S^* denotes the set of finite sequences of elements of S; S^ω the infinite sequences. Let $\zeta = s_0 s_1 \cdots$ be a (finite or infinite) sequence and $i \geq 0$. If ζ is finite of length n, assume $i < n$. Then $\zeta(i)$ denotes the element s_i. For any $i \geq 0$, ζ^i denotes the suffix $s_i s_{i+1} \cdots$. (ζ^i is empty if ζ is finite and $i \geq n$).

For $\zeta \in S^*$ and $\eta \in S^* \cup S^\omega$, $\zeta \circ \eta$ denotes the concatenation of ζ and η.

If $S \subseteq T$ and η is a sequence of elements of T, $\eta|_S$ denotes the sequence obtained by deleting from η all elements not in S.

2.1 Linear Temporal Logic

Let Act be a universal set of actions. We assume Act is infinite.

Definition 1. Form (the *LTL formulas over* Act) is the smallest set satisfying:

- $\mathsf{true} \in \mathsf{Form}$,
- if $a \in \mathsf{Act}$ then $a \in \mathsf{Form}$, and
- if f and g are in Form, so are $\neg f$, $f \wedge g$, $\mathbf{X} f$, and $f \mathbf{U} g$.

Additional operators are defined as shorthand for other formulas: $\mathsf{false} = \neg\mathsf{true}$, $f \vee g = \neg((\neg f) \wedge \neg g)$, $f \rightarrow g = (\neg f) \vee g$, $\mathbf{F} f = \mathsf{true}\mathbf{U} f$, $\mathbf{G} f = \neg\mathbf{F}\neg f$, and $f\mathbf{W}g = (f\mathbf{U}g) \vee \mathbf{G}f$. □

Definition 2. The *alphabet* of an LTL formula f, denoted αf, is the set of actions that occur syntactically within f. □

Definition 3. The *action-based semantics* of LTL is defined by the relation $\zeta \models_{\mathsf{A}} f$, where $\zeta \in \mathsf{Act}^\omega$ and $f \in \mathsf{Form}$, which is defined as follows:

- $\zeta \models_{\mathsf{A}} \mathsf{true}$,
- $\zeta \models_{\mathsf{A}} a$ iff $\zeta(0) = a$,
- $\zeta \models_{\mathsf{A}} \neg f$ iff $\zeta \not\models_{\mathsf{A}} f$,
- $\zeta \models_{\mathsf{A}} f \wedge g$ iff $\zeta \models_{\mathsf{A}} f$ and $\zeta \models_{\mathsf{A}} g$,
- $\zeta \models_{\mathsf{A}} \mathbf{X} f$ iff $\zeta^1 \models_{\mathsf{A}} f$, and
- $\zeta \models_{\mathsf{A}} f\mathbf{U}g$ iff $\exists i \geq 0 \,.\, (\zeta^i \models_{\mathsf{A}} g \;\wedge\; \forall j \in 0..i-1 \,.\, \zeta^j \models_{\mathsf{A}} f)$. □

When using the action-based semantics, the logic is sometimes referred to as "Action LTL" or ALTL [7,8].

The *state-based semantics* is defined by a relation $\xi \models_{\mathsf{S}} f$, where $\xi \in (2^{\mathsf{Act}})^\omega$. The definition of $\models_{\mathsf{S}}$ is well-known, and is exactly the same as Definition 3, except that $\xi \models_{\mathsf{S}} a$ iff $a \in \xi(0)$. The action semantics are consistent with the state semantics in the following sense. Let $f \in \mathsf{Form}$, and $\zeta = a_0 a_1 \cdots \in \mathsf{Act}^\omega$. Let $\xi = \{a_0\}\{a_1\} \cdots \in (2^{\mathsf{Act}})^\omega$. Then $\zeta \models_{\mathsf{A}} f$ iff $\xi \models_{\mathsf{S}} f$. The main difference between the state- and action-based formalisms is that in the state-based formalism, any number of atomic propositions can hold at each step. In the action-based formalism, precisely one action occurs in each step.

Definition 4. Let $f, g \in \mathsf{Form}$. Define

- (action equivalence) $f \equiv_{\mathsf{A}} g$ if $(\zeta \models_{\mathsf{A}} f \Leftrightarrow \zeta \models_{\mathsf{A}} g)$ for all $\zeta \in \mathsf{Act}^\omega$
- (state equivalence) $f \equiv_{\mathsf{S}} g$ if $(\xi \models_{\mathsf{S}} f \Leftrightarrow \xi \models_{\mathsf{S}} g)$ for all $\xi \in (2^{\mathsf{Act}})^\omega$. □

The following fact about the state-based semantics can be proved by induction on the formula structure:

Lemma 1. *Let $f \in \mathsf{Form}$ and $\xi = s_0 s_1 \cdots \in (2^{\mathsf{Act}})^\omega$. Let $\xi' = s'_0 s'_1 \cdots$, where $s'_i = \alpha f \cap s_i$. Then $\xi \models_{\mathsf{S}} f$ iff $\xi' \models_{\mathsf{S}} f$.*

The following shows that action LTL, like ordinary state-based LTL, is a decidable logic:

Proposition 1. *Let $f, g \in \mathsf{Form}$, $A = \alpha f \cup \alpha g$, and*

$$h = \mathbf{G}\Big[\big(\bigwedge_{a \in A} \neg a\big) \vee \bigvee_{a \in A} \big(a \wedge \bigwedge_{b \in A \setminus \{a\}} \neg b\big)\Big].$$

Then $f \equiv_{\mathsf{A}} g \Leftrightarrow f \wedge h \equiv_{\mathsf{S}} g \wedge h$. In particular, action equivalence is decidable.

Proof. Note the meaning of h: at each step in a state-based trace, at most one element of A is true.

Suppose $f \wedge h \equiv_{\mathsf{S}} g \wedge h$. Let $\zeta = a_0 a_1 \cdots \in \mathsf{Act}^\omega$. Let $\xi = \{a_0\}\{a_1\}\cdots$. We have $\xi \models h$. By the consistency of the state and action semantics, we have

$$\zeta \models_{\mathsf{A}} f \Leftrightarrow \xi \models_{\mathsf{S}} f \Leftrightarrow \xi \models_{\mathsf{S}} f \wedge h \Leftrightarrow \xi \models_{\mathsf{S}} g \wedge h \Leftrightarrow \xi \models_{\mathsf{S}} g \Leftrightarrow \zeta \models_{\mathsf{A}} g,$$

hence $f \equiv_{\mathsf{A}} g$.

Suppose instead that $f \equiv_{\mathsf{A}} g$. We wish to show $\xi \models_{\mathsf{S}} f \wedge h \Leftrightarrow \xi \models_{\mathsf{S}} g \wedge h$ for any $\xi = s_0 s_1 \cdots \in (2^{\mathsf{Act}})^\omega$. By Lemma 1, it suffices to assume $s_i \subseteq A$ for all i.

Let τ be any element of $\mathsf{Act} \setminus A$. (Here we are using the fact that Act is infinite, while A is finite.) If $|s_i| > 1$ for some i, then ξ violates h and therefore violates both $f \wedge h$ and $g \wedge h$. So suppose $|s_i| \leq 1$ for all i, which means $\xi \models_{\mathsf{S}} h$. Let $\zeta = a_0 a_1 \cdots$, where a_i is the sole member of s_i if $|s_i| = 1$, or τ if $|s_i| = 0$. By Lemma 1, $\xi \models_{\mathsf{S}} f$ iff $\{a_0\}\{a_1\}\cdots \models_{\mathsf{S}} f$. By the consistency of the action and state semantics, this is equivalent to $\zeta \models_{\mathsf{A}} f$. A similar statement holds for g. Hence

$$\xi \models_{\mathsf{S}} f \wedge h \Leftrightarrow \xi \models_{\mathsf{S}} f \Leftrightarrow \zeta \models_{\mathsf{A}} f \Leftrightarrow \zeta \models_{\mathsf{A}} g \Leftrightarrow \xi \models_{\mathsf{S}} g \Leftrightarrow \xi \models_{\mathsf{S}} g \wedge h.$$

The proposition reduces the question of action equivalence to one of ordinary (state) equivalence of LTL formulas, which is known to be decidable ([21], see also [30, Thm. 24]). □

Definition 5. For $A \subseteq \mathsf{Act}$ and $f \in \mathsf{Form}$ with $\alpha f \subseteq A$, let

$$\mathcal{L}(f, A) = \{\zeta \in A^\omega \mid \zeta \models f\}.$$ □

2.2 Büchi Automata

Definition 6. A *Büchi Automaton* (BA) over Act is a tuple $(S, \Sigma, \rightarrow, S^0, F)$ where

1. S is a finite set of *states*,
2. Σ, the *alphabet*, is a finite subset of Act,
3. $\rightarrow \subseteq S \times \Sigma \times S$ is the *transition relation*,
4. $S^0 \subseteq S$ is the set of *initial states*, and
5. $F \subseteq S$ is the set of *accepting states*. □

We will use the following notation and terminology for a BA B. The *source* of a transition (s, a, s') is s, the *destination* is s', and the *label* is a. We write $s \xrightarrow{a} s'$ as shorthand for $(s, a, s') \in \rightarrow$, and $s \xrightarrow{a_0 a_1 \ldots a_n} s'$ for $\exists s_1, s_2, \ldots s_n \in S \,.\, s \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots s_n \xrightarrow{a_n} s'$. For $a \in A$ and $s \in S$, we say a is *enabled at* s if $s \xrightarrow{a} s'$ for some $s' \in S$. The set of all actions enabled at s is denoted $\mathsf{enabled}(B, s)$.

For $s \in S$, a *path in* B *starting from* s is a (finite or infinite) sequence π of transitions such that (1) if π is not empty, the source of $\pi(0)$ is s, and (2) the destination of $\pi(i)$ is the source of $\pi(i+1)$ for all i for which these are defined. If π is not empty, define $\mathsf{first}(\pi)$ to be s; if π is finite, define $\mathsf{last}(\pi)$ to be the destination of the last transition of π. We say π *spells the word* $a_0 a_1 \cdots$, where a_i is the label of $\pi(i)$.

An infinite path is *accepting* if it visits a state in F infinitely often. An *(accepting) trace starting from* s is a word spelled by an (accepting) path starting from s. An *(accepting) trace of* B is an (accepting) trace starting from an initial state. The *language of* B, denoted $\mathcal{L}(B)$, is the set of all accepting traces of B.

Proposition 2. *There is an algorithm that consumes any finite subset A of Act and an $f \in \mathsf{Form}$ with $\alpha f \subseteq A$, and produces a BA B with alphabet A such that $\mathcal{L}(B) = \mathcal{L}(f, A)$.*

Proof. There are well-known algorithms to produce a BA C with alphabet 2^A which accepts exactly the words satisfying f under the state semantics (e.g., [31]). Let B be the same as C, except the alphabet is A and there is a transition $s \xrightarrow{a} s'$ in B iff there is a transition $s \xrightarrow{\{a\}} s'$ in C. We have

$$\begin{aligned} a_0 a_1 \cdots \in \mathcal{L}(B) &\Leftrightarrow \{a_0\}\{a_1\} \cdots \in \mathcal{L}(C) \\ &\Leftrightarrow \{a_0\}\{a_1\} \cdots \models_{\mathsf{s}} f \\ &\Leftrightarrow a_0 a_1 \cdots \in \mathcal{L}(f, A). \end{aligned}$$ □

In practice, tools that convert LTL formulas to BAs produce an automaton in which an edge is labeled by a propositional formula ϕ over αf. Such an edge represents a set of transitions, one for each $P \subseteq A$ for which ϕ holds for the valuation that assigns *true* to each element of P and *false* to each element of $A \setminus P$. In this case, the conversion to B entails creating one transition for each $a \in A$ for which ϕ holds when *true* is assigned to a and *false* is assigned to all other actions.

Definition 7. Let $B_i = (S_i, \Sigma_i, \to_i, S_i^0, F_i)$ $(i = 1, 2)$ denote two BAs over Act. The *parallel composition of* B_1 *and* B_2 is the BA

$$B_1 \parallel B_2 \equiv (S_1 \times S_2, \Sigma_1 \cup \Sigma_2, \to, S_1^0 \times S_2^0, F_1 \times F_2),$$

where $\to$ is defined by

$$\frac{s_1 \xrightarrow{a}_1 s_1' \quad a \notin \Sigma_2}{\langle s_1, s_2\rangle \xrightarrow{a} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{a}_2 s_2' \quad a \notin \Sigma_1}{\langle s_1, s_2\rangle \xrightarrow{a} \langle s_1, s_2'\rangle} \qquad \frac{s_1 \xrightarrow{a}_1 s_1' \quad s_2 \xrightarrow{a}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{a} \langle s_1', s_2'\rangle}. \qquad \square$$

If we flatten all tuples (e.g., identify $(S_1 \times S_2) \times S_3$ with $S_1 \times S_2 \times S_3$) then $\parallel$ is an associative operator.

Note that in the special case where the two automata have the same alphabet ($\Sigma_1 = \Sigma_2$), every action is synchronizing, and the parallel composition is the usual "synchronous product." In this case, $\mathcal{L}(B_1 \parallel B_2) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$.

2.3 Labeled Transition Systems

Definition 8. A *labeled transition system* (LTS) over Act is a tuple $(Q, A, \to, q^0)$ for which $(Q, A, \to, \{q^0\}, Q)$ is a BA over Act. In other words, it is a BA in which all states are accepting and there is only one initial state. $\square$

Definition 9. Let M be an LTS with alphabet A, and f an LTL formula with $\alpha f \subseteq A$. We write $M \models f$ if $\mathcal{L}(M) \subseteq \mathcal{L}(f, A)$. $\square$

The following observation is the basis of the automata-theoretic approach to model checking (cf. [30, §4.2]):

Proposition 3. *Let M be an LTS with alphabet A and f an LTL formula with $\alpha f \subseteq A$. Let B be a BA with $\mathcal{L}(B) = \mathcal{L}(\neg f, A)$. Then $M \models f \iff \mathcal{L}(M \parallel B) = \emptyset$.*

Proof. M and B have the same alphabet, so $\mathcal{L}(M \parallel B) = \mathcal{L}(M) \cap \mathcal{L}(B)$, hence

$$\mathcal{L}(M \parallel B) = \mathcal{L}(M) \cap \mathcal{L}(\neg f, A) = \mathcal{L}(M) \cap (A^\omega \setminus \mathcal{L}(f, A)) = \mathcal{L}(M) \setminus \mathcal{L}(f, A).$$

This set is empty iff $\mathcal{L}(M) \subseteq \mathcal{L}(f, A)$. $\square$

There are various algorithms to determine language emptiness of a BA; in this paper we will use the well-known Nested Depth First Search (NDFS) algorithm [2].

3 Interruptible Properties

An LTS comes with an alphabet, which is a subset A of Act. By a *property over* A we simply mean a subset P of A^ω. We say a trace $\zeta \in A^\omega$ *satisfies* P if $\zeta \in P$. We have already seen two ways to specify properties. An LTL formula f with $\alpha f \subseteq A$ specifies the property $\mathcal{L}(f, A)$. A Büchi automaton B with alphabet A specifies the property $\mathcal{L}(B)$. We next define a special class of properties:

Definition 10. Given sets $V \subseteq A \subseteq \mathsf{Act}$, we say a property P over A is V-*interruptible* if

$$\zeta|_V = \eta|_V \Rightarrow (\zeta \in P \Leftrightarrow \eta \in P) \qquad \text{for all } \zeta, \eta \in A^\omega.$$

An LTL formula f is V-*interruptible* if $\mathcal{L}(f, \mathsf{Act})$ is V-interruptible. We say f is *interruptible* if f is αf-interruptible. The set of all interruptible LTL formulas is denoted Intrpt. □

The set V is known as the *visible set*. The definition essentially says that the insertion or deletion of invisible actions (those in $A \backslash V$) has no bearing on whether a trace satisfies P. Put another way, the question of whether a trace belongs to P is determined purely by its visible actions. The following collects some basic facts about interruptibility. All follow immediately from the definitions.

Proposition 4. *Let $V \subseteq A \subseteq \mathsf{Act}$, $P \subseteq A^\omega$ and $f, g \in \mathsf{Form}$. Then all of the following hold:*

1. *P is A-interruptible.*
2. *If P is V-interruptible, and $V \subseteq V'$, then P is V'-interruptible.*
3. *If f is interruptible and $\alpha f \subseteq A$, then $\mathcal{L}(f, A)$ is αf-interruptible.*
4. *f is interruptible iff the following holds:*

$$\forall \zeta, \eta \in \mathsf{Act}^\omega \,.\, (\zeta|_{\alpha f} = \eta|_{\alpha f} \wedge \zeta \models_{\mathsf{A}} f) \Rightarrow \eta \models_{\mathsf{A}} f.$$

5. *If $\alpha f = \alpha g$ and $f \equiv_{\mathsf{A}} g$ then f is interruptible iff g is interruptible.*

Lemma 2. *Suppose $V \subseteq A \subseteq \mathsf{Act}$ and P_1 and P_2 are V-interruptible properties over A. Let $\mathcal{F} = V^\omega \cup V^* \circ (A \setminus V)^\omega$. Then $P_1 = P_2$ iff $P_1 \cap \mathcal{F} = P_2 \cap \mathcal{F}$.*

Proof. Assume $P_1 \cap \mathcal{F} = P_2 \cap \mathcal{F}$. Let $\zeta \in P_1$. If $\zeta|_V$ is infinite, then since $\zeta|_V|_V = \zeta|_V$, and P_1 is V-interruptible, $\zeta|_V \in P_1$. But $\zeta|_V \in V^\omega$, so $\zeta|_V \in P_1 \cap \mathcal{F}$, and therefore $\zeta|_V \in P_2$. Since P_2 is V-interruptible, $\zeta \in P_2$.

If $\zeta|_V$ is finite, there is a prefix θ of ζ such that $\zeta = \theta \circ \eta$, with $\eta \in (V \setminus A)^\omega$. Let $\xi = \theta|_V \circ \eta$. We have $\xi \in V^* \circ (A \setminus V)^\omega$ and $\xi|_V = \zeta|_V$, hence $\xi \in P_1 \cap \mathcal{F}$. Therefore $\xi \in P_2$, and since P_2 is V-interruptible, $\zeta \in P_2$. □

The elements of $\mathcal{F}$ are known as the V-*interrupt-free* words over A.

3.1 Decidability of Interruptibility of LTL Formulas

We next show that interruptibility is a decidable property of LTL formulas. Define $\mathsf{intrpt}\colon \mathsf{Form} \to \mathsf{Form}$ as follows. Given $f \in \mathsf{Form}$, let $V = \alpha f$ and $\hat{V} = \bigvee_{a \in V} a$, and define $\beta\colon \mathsf{Form} \to \mathsf{Form}$ by

$$\begin{aligned}
\beta(\mathsf{true}) &= \mathsf{true}\\
\beta(a) &= (\neg\hat{V})\mathbf{U}a\\
\beta(\neg f_1) &= \neg\beta(f_1)\\
\beta(f_1 \wedge f_2) &= \beta(f_1) \wedge \beta(f_2)\\
\beta(\mathbf{X}f_1) &= ((\neg\hat{V})\mathbf{U}(\hat{V} \wedge \mathbf{X}\beta(f_1))) \,\vee\, ((\mathbf{G}\neg\hat{V}) \wedge \mathbf{X}\beta(f_1))\\
\beta(f_1\mathbf{U}f_2) &= \beta(f_1)\mathbf{U}\beta(f_2).
\end{aligned}$$

for $a \in \mathsf{Act}$ and $f_1, f_2 \in \mathsf{Form}$. Let $\mathsf{intrpt}(f) = \beta(f)$.

Theorem 1. *Let f be an LTL formula over* Act. *The following hold:*

1. $\mathsf{intrpt}(f)$ *is interruptible.*
2. f *is interruptible iff* $\mathsf{intrpt}(f) \equiv_{\mathsf{A}} f$.

In particular, interruptibility of LTL formulas is decidable.

Before proving Theorem 1, we give some intuition regarding the definition of intrpt. Function β can be thought of as consuming a property on V-interrupt-free words (i.e., words in $V^\omega \cup V^* \circ (A \setminus V)^\omega$) and extending it to a property on all words (A^ω). It is designed so that $\beta(g)$ is V-interruptible and agrees with g on V-interrupt-free words. For example, the formula a means "a is the first action" (in an interrupt-free word), which extends to the property "a is the first visible action" (in an arbitrary word). The formula $\mathbf{X}f_1$ states "f_1 holds after removing the first action," so $\beta(\mathbf{X}f_1)$ should declare "$\beta(f_1)$ holds after removing the prefix ending in the first visible action." That is almost correct, but there is also the possibility that an element of A^ω has no visible action, which is the reason for the second clause in the definition of $\beta(\mathbf{X}f_1)$.

The remainder of this section is devoted to the proof of Theorem 1. First note that $\mathsf{intrpt}(f)$ and f have the same alphabet, i.e., $\alpha\mathsf{intrpt}(f) = V$.

Proof of part 1. Say a subformula g of f is *good* if $\beta(g)$ is V-interruptible, i.e.,

$$\forall \zeta, \eta \in \mathsf{Act}^\omega \,.\, \zeta|_V = \eta|_V \Rightarrow (\zeta \models_{\mathsf{A}} \beta(g) \;\Leftrightarrow\; \eta \models_{\mathsf{A}} \beta(g)).$$

We show by induction on formula structure that every subformula of f is good. The case $g = f$ will show that $\mathsf{intrpt}(f)$ is interruptible. Assume throughout that $\zeta|_V = \eta|_V$.

If $g = \mathsf{true}$ then $\beta(g) = \mathsf{true}$, so g is clearly good.

If $g = a$ for some $a \in \mathsf{Act}$, then $\zeta \models_{\mathsf{A}} \beta(g) = (\neg\hat{V})\mathbf{U}a$ iff $\zeta|_V$ is non-empty and $\mathsf{first}(\zeta|_V) = a$. Since this depends only on $\zeta|_V$, g is good.

If $g = \neg f_1$ and f_1 is good, then g is good because

$$\zeta \models_{\mathsf{A}} \beta(g) \;\Leftrightarrow\; \zeta \not\models_{\mathsf{A}} \beta(f_1) \;\Leftrightarrow\; \eta \not\models \beta(f_1) \;\Leftrightarrow\; \eta \models_{\mathsf{A}} \beta(g).$$

If $g = f_1 \wedge f_2$, and f_1 and f_2 are good, then g is good because

$$\begin{aligned} \zeta \models_{\mathsf{A}} \beta(g) &\;\Leftrightarrow\; \zeta \models_{\mathsf{A}} \beta(f_1) \,\wedge\, \zeta \models_{\mathsf{A}} \beta(f_2) \\ &\;\Leftrightarrow\; \eta \models_{\mathsf{A}} \beta(f_1) \,\wedge\, \eta \models_{\mathsf{A}} \beta(f_2) \;\Leftrightarrow\; \eta \models_{\mathsf{A}} \beta(g). \end{aligned}$$

Suppose $g = \mathbf{X}f_1$ and f_1 is good. There are two cases:

- **Case 1:** $\zeta|_V$ is empty. Then no suffix of ζ or η satisfies $\hat{V}$. Hence

$$\theta \models_{\mathsf{A}} \beta(g) \;\Leftrightarrow\; \theta \models_{\mathsf{A}} \mathbf{X}\beta(f_1) \;\Leftrightarrow\; \theta^1 \models_{\mathsf{A}} \beta(f_1) \qquad (\theta \in \{\zeta, \eta\}).$$

  Moreover, $\zeta^1|_V = \eta^1|_V$ (as both are empty), and $\beta(f_1)$ is good, so we have $\zeta^1 \models_{\mathsf{A}} \beta(f_1) \;\Leftrightarrow\; \eta^1 \models_{\mathsf{A}} \beta(f_1)$. These show $\zeta \models_{\mathsf{A}} \beta(g) \;\Leftrightarrow\; \eta \models_{\mathsf{A}} \beta(g)$.

– **Case 2:** $\zeta|_V$ is nonempty. Let i be the index of the first occurrence of an element of V in ζ, and j the similar index for η. We have

$$\zeta^{i+1}|_V = (\zeta|_V)^1 = (\eta|_V)^1 = \eta^{j+1}|_V.$$

As f_1 is good, it follows that $\zeta^{i+1} \models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \eta^{j+1} \models_{\mathsf{A}} \beta(f_1)$. Hence

$$\zeta \models_{\mathsf{A}} \beta(g) \Leftrightarrow \zeta^{i+1} \models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \eta^{j+1} \models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \eta \models_{\mathsf{A}} \beta(g).$$

Suppose $g = f_1 \mathbf{U} f_2$ and f_1 and f_2 are good. We have $\beta(g) = \beta(f_1)\mathbf{U}\beta(f_2)$. If $\zeta \models_{\mathsf{A}} \beta(g)$ then there exists $i \geq 0$ such that $\zeta^i \models_{\mathsf{A}} \beta(f_2)$ and $\zeta^j \models_{\mathsf{A}} \beta(f_1)$ for $j < i$. Now there is some $i' \geq 0$ such that $\eta^{i'}|_V = \zeta^i|_V$ and for all $j' < i'$, there is some $j < i$ such that $\eta^{j'}|_V = \zeta^j|_V$. It follows that $\eta \models \beta(g)$. Hence g is good.

Proof of part 2. Suppose first that $\mathsf{intrpt}(f) \equiv_{\mathsf{A}} f$. From part 1, $\mathsf{intrpt}(f)$ is interruptible, so Proposition 4(5) implies f is interruptible.

Suppose instead that f is interruptible. We wish to show $\mathsf{intrpt}(f) \equiv_{\mathsf{A}} f$. By Lemma 2, it suffices to show the two formulas agree on V-interrupt-free words. We will show by induction that for each subformula g of f, $\zeta \models_{\mathsf{A}} g \Leftrightarrow \zeta \models_{\mathsf{A}} \beta(g)$ for all V-interrupt-free ζ. The case $g = f$ will complete the proof.

If $g = \mathsf{true}$, $\beta(g) = \mathsf{true}$ and the condition clearly holds.

If $g = a$ for some $a \in \mathsf{Act}$, $\zeta \models_{\mathsf{A}} \beta(g) \Leftrightarrow \zeta \models_{\mathsf{A}} (\neg\hat{V})\mathbf{U}a \Leftrightarrow \zeta \models_{\mathsf{A}} a$, as ζ is V-interrupt-free.

If $g = \neg f_1$ and the inductive hypothesis holds for f_1, then

$$\zeta \models_{\mathsf{A}} \beta(g) \Leftrightarrow \zeta \not\models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \zeta \not\models_{\mathsf{A}} f_1 \Leftrightarrow \zeta \models_{\mathsf{A}} g.$$

If $g = f_1 \wedge f_2$ and the inductive hypothesis holds for f_1 and f_2 then

$$\zeta \models_{\mathsf{A}} \beta(g) \Leftrightarrow \zeta \models_{\mathsf{A}} \beta(f_1) \wedge \zeta \models_{\mathsf{A}} \beta(f_2) \Leftrightarrow \zeta \models_{\mathsf{A}} f_1 \wedge \zeta \models_{\mathsf{A}} f_2 \Leftrightarrow \zeta \models_{\mathsf{A}} g.$$

Suppose $g = \mathbf{X}f_1$ and the inductive hypothesis holds for f_1. Note that any suffix of a V-interrupt-free word, e.g., ζ^1, is also V-interrupt-free. If $\zeta|_V$ is empty,

$$\zeta \models_{\mathsf{A}} \beta(g) \Leftrightarrow \zeta \models_{\mathsf{A}} \mathbf{X}\beta(f_1) \Leftrightarrow \zeta^1 \models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \zeta^1 \models_{\mathsf{A}} f_1 \Leftrightarrow \zeta \models_{\mathsf{A}} g.$$

If $\zeta|_V$ is nonempty, then $\zeta \models_{\mathsf{A}} \hat{V}$, so

$$\begin{aligned}\zeta \models_{\mathsf{A}} \beta(g) &\Leftrightarrow \zeta \models_{\mathsf{A}} (\neg\hat{V})\mathbf{U}(\hat{V} \wedge \mathbf{X}\beta(f_1)) \Leftrightarrow \zeta \models_{\mathsf{A}} \mathbf{X}\beta(f_1)\\ &\Leftrightarrow \zeta^1 \models_{\mathsf{A}} \beta(f_1) \Leftrightarrow \zeta^1 \models_{\mathsf{A}} f_1 \Leftrightarrow \zeta \models_{\mathsf{A}} g.\end{aligned}$$

If $g = f_1\mathbf{U}f_2$, then applying the inductive hypothesis to f_1 and f_2 yields

$$\begin{aligned}\zeta \models_{\mathsf{A}} g &\Leftrightarrow \exists i > 0 \,.\, \zeta^i \models_{\mathsf{A}} f_2 \wedge \forall j < i \,.\, \zeta^j \models_{\mathsf{A}} f_1\\ &\Leftrightarrow \exists i > 0 \,.\, \zeta^i \models_{\mathsf{A}} \beta(f_2) \wedge \forall j < i \,.\, \zeta^j \models_{\mathsf{A}} \beta(f_1)\\ &\Leftrightarrow \zeta \models_{\mathsf{A}} \beta(g).\end{aligned}$$

Decidability follows from part 2 and Proposition 1.

3.2 Examples of Interruptible LTL Formulas

We now give some examples of LTL formulas that are, and are not, interruptible. Suppose $a, b \in \mathsf{Act}$ and $a \neq b$. The formula $\mathbf{G}a$ is not interruptible, because $\zeta = a^\omega$ satisfies $\mathbf{G}a$ but $\eta = ba^\omega$ does not, even though $\zeta|_{\{a\}} = \eta|_{\{a\}}$. On the other hand, $\mathbf{G}\neg a$ is interruptible, since if two words have the same restriction to $\{a\}$, then one will contain an a if, and only if, the other contains an a. Similarly, $\mathbf{F}a$ is interruptible and $\mathbf{F}\neg a$ is not. The formula $\mathbf{F}(a \wedge \mathbf{XF}a)$ is also interruptible and means "a occurs at least twice."

Proposition 5. *There exist* $\mathsf{Pos}, \mathsf{Neg} \subseteq \mathsf{Form}$ *such that (i) for all* $f, f' \in \mathsf{Form}$*,*

$$(f \in \mathsf{Pos} \wedge f' \equiv_{\mathsf{A}} f) \Rightarrow f' \in \mathsf{Pos}$$
$$(f \in \mathsf{Neg} \wedge f' \equiv_{\mathsf{A}} f) \Rightarrow f' \in \mathsf{Neg},$$

and (ii) for all $a \in \mathsf{Act}$*,* $f_1, f_2 \in \mathsf{Intrpt}$*,* $g_1, g_2 \in \mathsf{Pos}$*, and* $h_1, h_2 \in \mathsf{Neg}$*,*

$$\begin{aligned} \mathit{false},\ a,\ \neg h_1,\ g_1 \wedge g_2,\ g_1 \vee g_2,\ a \wedge f_1,\ a \wedge \mathbf{X}f_1\ &\in\ \mathsf{Pos} \\ \mathit{true},\ \neg a,\ \neg g_1,\ h_1 \wedge h_2,\ h_1 \vee h_2,\ \neg a \vee f_1,\ \neg a \vee \mathbf{X}f_1\ &\in\ \mathsf{Neg} \\ \mathit{true},\ \mathit{false},\ f_1 \wedge f_2,\ f_1 \vee f_2,\ \neg f_1,\ \mathbf{F}g_1,\ \mathbf{G}h_1,\ f_1\mathbf{U}f_2,\ h_1\mathbf{U}g_1,\ h_1\mathbf{U}f_1\ &\in\ \mathsf{Intrpt}. \end{aligned}$$

Proposition 5 can be used to generate many interruptible formulas. For example, $\mathbf{G}(a \to \mathbf{F}b)$, which occurs multiple times in the RERS suite, is seen to be interruptible as follows. First $b \in \mathsf{Pos}$, so by the proposition, $\mathbf{F}b \in \mathsf{Intrpt}$. Again by the proposition, $\neg a \vee \mathbf{F}b \in \mathsf{Neg}$. Since this last formula is equivalent to $a \to \mathbf{F}b$, we have $a \to \mathbf{F}b \in \mathsf{Neg}$. Therefore $\mathbf{G}(a \to \mathbf{F}b) \in \mathsf{Intrpt}$. The proof of the proposition is included in the Appendix.

Note that Intrpt and the set of stutter-invariant formulas are not comparable. For example, $f = \mathbf{F}(a \wedge \mathbf{XF}a)$ is interruptible, but not stutter-invariant. In fact f is not action-equivalent to any stutter-invariant formula g, since if there were such a g, the sequence aab^ω would satisfy g, but the stutter-equivalent sequence ab^ω cannot satisfy g. Conversely, the formulas a and $\mathbf{G}a$ are both stutter-invariant, but neither is interruptible. The formula $\mathbf{F}a$ is both stutter-invariant and interruptible. Finally, the formula $\mathbf{X}a$ is neither stutter-invariant nor interruptible.

3.3 Decidability of Interruptibility of Büchi Automata

Definition 11. Let B be a BA with alphabet A, $V \subseteq A$ (the *visible* actions), and $I = A \setminus V$ (the *invisible actions*). We say B is in *V-interrupt normal form* if the following hold for any $x \in I$, $a \in A$, and states s_1, s_2, and s_3:

1. If $s_1 \xrightarrow{a} s_2$ then B has a state s_1' such that $s_1 \xrightarrow{x} s_1' \xrightarrow{a} s_2$.
2. If $s_1 \xrightarrow{x} s_2 \xrightarrow{a} s_3$ then $s_1 \xrightarrow{a} s_3$ and if s_2 is accepting then s_1 or s_3 is accepting.
3. If $s_1 \xrightarrow{x} s_2$ then $s_1 \xrightarrow{y} s_2$ for all $y \in I$.

Proposition 6. *Suppose B is in V-interrupt normal form. Then $\mathcal{L}(B)$ is V-interruptible.*

Proof. Suppose $\zeta, \eta \in A^\omega$, $\zeta \in \mathcal{L}(B)$, and $\zeta|_V = \eta|_V$. We wish to show $\eta \in \mathcal{L}(B)$. Let π be an accepting path for ζ.

Assume $\zeta|_V$ is infinite. By Definition 11(2), we can remove all invisible transitions from the accepting path π, and the result is an accepting path that spells $\zeta|_V$. By Definition 11(1), we can insert any arbitrary finite sequence of invisible transition between two consecutive visible transitions; we can therefore construct an accepting path for η.

If $\zeta|_V$ is finite, proceed as above to form an accepting path which spells a finite prefix of η followed by an infinite word of invisible actions. By Definition 11(3), that infinite suffix can be transformed to spell any infinite word of invisibles, and in that way one obtains an accepting path for η. ☐

Given any BA $B = (S, A, T, S^0, F)$ and a visible set $V \subseteq A$, define a BA $\mathsf{norm}(B, V)$ as follows: if $V = A$, $\mathsf{norm}(B, V) = B$, otherwise $\mathsf{norm}(B, V)$ is $\hat{B} = (\hat{S}, A, \hat{T}, \hat{S}^0, \hat{F})$, where

$$\begin{aligned}
D &= \{s \in S \mid \text{there is an accepting path from } s \text{ with all labels in } I\}\\
\hat{S} &= \{\hat{u} \mid u \in S\} \cup \{u^\sharp \mid u \in F \setminus D\} \cup \{\mathsf{DIV}\}\\
\hat{S}^0 &= \{\hat{u} \mid u \in S^0\}\\
\hat{F} &= \{\hat{u} \mid u \in F\} \cup \{\mathsf{DIV}\}\\
\hat{T} &= \{(\hat{u}, a, \hat{v}) \mid a \in V \wedge u, v \in S \wedge (u, a, v) \in T\} \cup\\
&\quad\ \{(\hat{u}, x, \hat{u}) \mid x \in I \wedge u \in D \cup (S \setminus F)\} \cup\\
&\quad\ \{(\mathsf{DIV}, x, \mathsf{DIV}) \mid x \in I\} \cup\\
&\quad\ \{(\hat{u}, x, \mathsf{DIV}) \mid x \in I \wedge u \in D \setminus F\} \cup\\
&\quad\ \{(\hat{u}, x, u^\sharp), (u^\sharp, x, u^\sharp) \mid x \in I \wedge u \in F \setminus D\} \cup\\
&\quad\ \{(u^\sharp, a, \hat{v}) \mid a \in V \wedge u \in F \setminus D \wedge v \in S \wedge (u, a, v) \in T\}
\end{aligned}$$

The set $\hat{S}$ consists of the *original states* $\hat{u}$, the *sharp states* $u^\sharp$, and one additional state DIV. The mapping from S to $\hat{S}$ defined by $u \mapsto \hat{u}$ is injective and preserves acceptability and visible transitions, i.e., for any $u, v \in S$ and $a \in V$, $u \xrightarrow{a} v \iff \hat{u} \xrightarrow{a} \hat{v}$. It follows that paths in B in which all labels are visible correspond one-to-one with paths through original states in $\hat{B}$ in which all labels are visible. Note that every invisible transition in $\hat{B}$ is a self-loop or ends in a sharp state or DIV. Moreover, all transitions in $\hat{B}$ ending in a sharp state or DIV are invisible.

Proposition 7. *For any BA B with alphabet A, and any visible set $V \subseteq A$, $\mathsf{norm}(B, V)$ is in V-interruptible normal form.*

Proof. To see Definition 11(1), suppose $s_1 \xrightarrow{a} s_2$. If $s_1 \xrightarrow{x} s_1$, take $s_1' = s_1$. Otherwise, $s_1 = \hat{u}$ for some $u \in F \setminus D$, and we can take $s_1' = u^\sharp$.

For Definition 11(2), suppose $s_1 \xrightarrow{x} s_2 \xrightarrow{a} s_3$. We need to show $s_1 \xrightarrow{a} s_3$ and if s_2 is accepting then s_1 or s_3 is accepting. If $s_1 = s_2$, the result is clear, so assume $s_1 \neq s_2$. There are then two cases: $s_2 = \mathsf{DIV}$ or $s_2 = u^\sharp$ for some $u \in F \setminus D$.

If $s_2 = \mathsf{DIV}$, then $a \in I$ and $s_3 = \mathsf{DIV}$, and we have $s_1 \xrightarrow{a} \mathsf{DIV}$. As DIV is accepting, the desired conclusion holds.

If $s_2 = u^\sharp$, then $s_1 = \hat{u}$, which is accepting. There are again two cases: either $s_3 = u^\sharp$ or $s_3 = \hat{v}$ for some $v \in S$. If $s_3 = u^\sharp$ then $a \in I$ and $\hat{u} \xrightarrow{a} u^\sharp$, as required. If $s_3 = \hat{v}$, then $a \in V$ and therefore $u \xrightarrow{a} v$, hence $\hat{u} \xrightarrow{a} \hat{v}$, as required.

Definition 11(3) is clear from the definition of $\hat{T}$. □

Theorem 2. *$\mathcal{L}(B)$ is V-interruptible iff $\mathcal{L}(\mathsf{norm}(B,V)) = \mathcal{L}(B)$. In particular interruptibility for Büchi Automata is decidable.*

Proof. Let $P_1 = \mathcal{L}(B)$ and $P_2 = \mathcal{L}(\mathsf{norm}(B,V))$. By Proposition 7, $\mathsf{norm}(B,V)$ is in V-interruptible normal form, so by Proposition 6, P_2 is V-interruptible. Hence one direction is clear: if $P_1 = P_2$, then P_1 is V-interruptible.

So suppose P_1 is V-interruptible. We wish to show $P_1 = P_2$. By Lemma 2, it suffices to show the two languages contain the same V-interrupt-free words.

Suppose ζ is a V-interrupt-free word in P_1. If $\zeta \in V^\omega$ then an accepting path θ in B maps to the accepting path $\hat{\theta}$ in $\hat{B}$, and $\zeta \in P_2$. So assume $\zeta \in V^* I^\omega$. Then an accepting path in B has a prefix θ of visible transitions ending in a state $u \in D$. That prefix corresponds to a path $\hat{\theta}$ in $\hat{B}$ ending in $\hat{u}$. As $u \in D$, $\hat{u} \xrightarrow{x} \hat{u}$ for all $x \in I$. If u is accepting, we get an accepting path for ζ that follows $\hat{\theta}$ and then loops at $\hat{u}$. If u is not accepting then $u \in D \setminus F$, and $\hat{u} \xrightarrow{x} \mathsf{DIV}$ for all $x \in I$. Since DIV is accepting and $\mathsf{DIV} \xrightarrow{x} \mathsf{DIV}$ for all $x \in I$, we again get an accepting path for ζ in $\hat{B}$.

Suppose now that ζ is a V-interrupt-free word in P_2. Assume $\zeta \in V^\omega$. An accepting path for ζ cannot pass through a sharp state or DIV, because only invisible transitions end in those states. So the path passes through only original states, and therefore corresponds to an accepting path in B.

Suppose $\zeta \in V^* I^\omega$. An accepting path for ζ in $\hat{B}$ consists of a prefix $\hat{\theta}$ of visible transitions followed by an infinite accepting path ξ of invisible transitions. As above, $\hat{\theta}$ corresponds to a path θ in B ending in a state u.

We claim that ξ cannot pass through a sharp state. This is because all invisible transitions departing from a sharp state are self loops. But sharp states are not accepting, while ξ is an accepting path of invisible transitions. It follows that each transition in ξ is a self-loop or terminates in DIV.

We now claim $u \in D$. For suppose the first transition in ξ is a self-loop on $\hat{u}$. According to the definition of $\hat{T}$, this implies $u \in D \cup (S \setminus F)$. Hence, if $u \notin D$ then u is not accepting, and all invisible transitions departing from $\hat{u}$ are self-loops, contradicting the fact that ξ is an accepting path. If, on the other hand, the first transition in ξ is $\hat{u} \xrightarrow{x} \mathsf{DIV}$, for some $x \in I$, then the definition of $\hat{T}$ implies $u \in D$, establishing the claim.

So $u \in D$, i.e., there is an accepting path ρ in B starting from u and consisting of all invisible transitions. The accepting path obtained by concatenating θ and ρ spells a word which, projected onto V, equals $\zeta|_V$. Since P_1 is V-interruptible, $\zeta \in P_1$. This completes the proof that $P_1 = P_2$.

The theorem reduces the problem of determining V-interruptibility to a problem of determining the equivalence of two Büchi Automata, which can be done, e.g., using language intersection, complement, and emptiness algorithms for BAs [31]. □

4 On-the-fly Partial Order Reduction

4.1 General Theory and Soundness Theorem

Let $M = (Q, A, T, q^0)$ be an LTS, $V \subseteq A$, and $B = (S, A, \delta, S^0, F)$ a V-interruptible BA. The goal of on-the-fly POR is to explore a subset R' of $R = M \parallel B$ with the property that $\mathcal{L}(R) = \emptyset \Leftrightarrow \mathcal{L}(R') = \emptyset$.

A function $\mathsf{amp}\colon Q \times S \to 2^A$ is an *ample selector* if $\mathsf{amp}(q, s) \subseteq \mathsf{enabled}(M, q)$ for all $q \in Q, s \in S$. Each $\mathsf{amp}(q, s)$ is an *ample set*. An ample selector determines a BA $R' = \mathsf{reduced}(R, \mathsf{amp})$ which has the same states, accepting states, and initial state as R, but only a subset of the transitions:

$$R' = (Q \times S, A, \delta', \{q^0\} \times S^0, Q \times F)$$
$$\delta' = \{((q, s), a, (q', s')) \mid a \in \mathsf{amp}(q, s) \wedge (q, a, q') \in T \wedge (s, a, s') \in \delta\}.$$

We now define some constraints on an ample selector that will be used to guarantee the reduced product space has nonempty language if the full space does. First we need the usual notion of independence:

Definition 12. Let M be an LTS with alphabet A, and $a, b \in A$. We say a and b are *independent* if all of the following hold for all states q and q' of M:

1. $q \xrightarrow{a} q' \Rightarrow b \in \mathsf{enabled}(M, q')$
2. $q \xrightarrow{b} q' \Rightarrow a \in \mathsf{enabled}(M, q')$
3. $q \xrightarrow{ab} q' \;\Leftrightarrow\; q \xrightarrow{ba} q'$.

We say a and b are *dependent* if they are not independent. □

Note that, in contrast with [1], we do not assume actions are deterministic. We can now define the four constraints:

C0 For all $q \in Q$, $s \in S$: $\mathsf{enabled}(M, q) \neq \emptyset \Rightarrow \mathsf{amp}(q, s) \neq \emptyset$.
C1 For all $q \in Q$, $s \in S$: on any trace in M starting from q, no action outside of $\mathsf{amp}(q, s)$ but dependent on an action in $\mathsf{amp}(q, s)$ can occur without an action in $\mathsf{amp}(q, s)$ occurring first.
C2 For all $q \in Q$, $s \in S$: if $\mathsf{amp}(q, s) \neq \mathsf{enabled}(M, q)$, then $\mathsf{amp}(q, s) \cap V = \emptyset$.
C3 For all $a \in A$: on any cycle in R' for which a is enabled in R at each state, there is some state σ on the cycle for which $a \in \mathsf{amp}(\sigma)$.

Theorem 3. *Let M be an LTS with alphabet A, $V \subseteq A$, B a BA with alphabet A in V-interruptible normal form, $R = M \parallel B$, and* amp *an ample selector satisfying* **C0**–**C3**. *Then $\mathcal{L}(\mathsf{reduced}(R, \mathsf{amp})) = \emptyset \Leftrightarrow \mathcal{L}(R) = \emptyset$.*

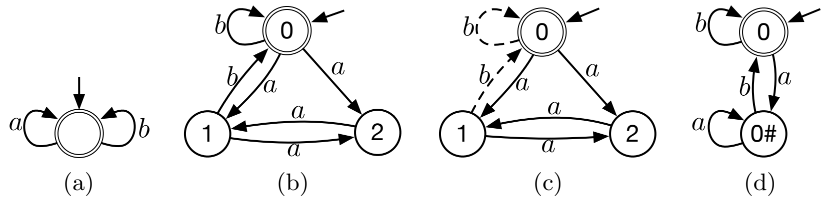

Fig. 1. Counterexample to Theorem 3 if B is not in interrupt normal form: (a) the LTS M, (b) the BA B representing $\mathbf{GF}b$, (c) the product space—dashed edges are in the full, but not reduced, space, and (d) the result of normalizing B and removing unreachable states.

The requirement that B be in interruptible normal form is necessary. A counterexample when that condition is not met is given in Figure 1. Note a and b are independent, and a is invisible. The ample set for product states 0 and 1 is $\{a\}$; the ample set for product state 2 is $\{a, b\}$. Hence **C3** holds because a state on the sole cycle is fully enabled. After normalizing B (and removing unreachable states), this problem goes away.

The remainder of this section is devoted to the proof of Theorem 3. The proof follows the structure of the proof in the state-based case [22], but there are subtle adjustments needed to adapt it to the action formalism.

Let θ be an accepting path in R. An infinite sequence of accepting paths $\pi_0, \pi_1, \ldots$ will be constructed, where $\pi_0 = \theta$. For each $i \geq 0$, π_i will be decomposed as $\eta_i \circ \theta_i$, where η_i is a finite path of length i in R', θ_i is an infinite path, and η_i is a prefix of η_{i+1}. For $i = 0$, η_0 is empty and $\theta_0 = \theta$.

Assume $i \geq 0$ and we have defined η_j and θ_j for $j \leq i$. Write

$$\theta_i \;=\; \langle q_0, s_0\rangle \xrightarrow{a_1} \langle q_1, s_1\rangle \xrightarrow{a_2} \cdots \tag{1}$$

Then η_{i+1} and θ_{i+1} are defined as follows. Let $E = \mathsf{amp}(q_0, s_0)$. There are two cases:

Case 1: $a_1 \in E$. Let η_{i+1} be the path obtained by appending the first transition of θ_i to η_i, and θ_{i+1} the path obtained by removing the first transition from θ_i.

Case 2: $a_1 \notin E$. Then there are two sub-cases:

Case 2a: Some operation in E occurs in θ_i. Let n be the index of the first such occurrence. By **C1**, a_j and a_n are independent for $1 \leq j < n$. By repeated application of the independence property, there is a path in M of the form

$$q_0 \overset{a_n}{\to} q'_1 \overset{a_1}{\to} q'_2 \overset{a_2}{\to} \cdots \overset{a_{n-2}}{\to} q'_{n-1} \overset{a_{n-1}}{\to} q_n \overset{a_{n+1}}{\to} q_{n+1} \overset{a_{n+2}}{\to} \cdots .$$

By **C2**, a_n is invisible. By Definition 11, B has an accepting path of the form

$$s_0 \overset{a_n}{\to} s'_0 \overset{a_1}{\to} s_1 \overset{a_2}{\to} \cdots \overset{a_{n-2}}{\to} s_{n-2} \overset{a_{n-1}}{\to} s_{n-1} \overset{a_{n+1}}{\to} s_{n+1} \overset{a_{n+2}}{\to} \cdots .$$

Composing these two paths yields a path in R. Removing the first transition (labeled a_n) yields θ_{i+1}. Appending that transition to η_i yields η_{i+1}.

Case 2b: No operation in E occurs in θ_i. By **C0**, E is nonempty. Let $b \in E$. By **C2**, every action in θ_i is independent of b. As in the case above, we obtain a path in R

$$\langle q_0, s_0 \rangle \xrightarrow{b} \langle q_1', s_0' \rangle \xrightarrow{a_1} \langle q_2', s_1 \rangle \xrightarrow{a_2} \langle q_3', s_2 \rangle \xrightarrow{a_3} \cdots .$$

and define θ_{i+1} and η_{i+1} as above.

Let η be the limit of the η_i, i.e., $\eta(i) = \eta_{i+1}(i)$. It is clear that η is an infinite path in R', but we must show it passes through an accepting state infinitely often. To see this, define integers d_i for $i \geq 0$ as follows. Let $\xi_i = s_0 s_1 \cdots$ be the sequence of BA states traced by θ_i. Let d_i be the minimum $j \geq 0$ such that s_j is accepting. Note that $d_i = 0$ iff $\mathsf{last}(\eta_i)$ is accepting.

Suppose $i \geq 0$ and $d_i > 0$. If Case 1 holds, then $d_{i+1} = d_i - 1$, since $\xi_{i+1} = \xi_i^1$. It is not hard to see that if Case 2 holds, $d_{i+1} \leq d_i$. Note that in Case 2a, if $d_i = n$, the accepting state s_n is removed, but Definition 11(2) guarantees that at least one of s_{n-1} and s_{n+1} is accepting. In the worst case (s_{n-1} is not accepting), we still have $d_{i+1} = n$.

We claim there are an infinite number of $i \geq 0$ such that Case 1 holds. Otherwise, there is some $i > 0$ such that Case 2 holds for all $j \geq i$. Let a be the first action in θ_i. Then for all $j \geq i$, a is the first action of θ_j and a is not in the ample set of $\mathsf{last}(\eta_j)$. Since the number of states of R is finite, there is some $k > i$ such that $\mathsf{last}(\eta_k) = \mathsf{last}(\eta_i)$. Hence there is a cycle in R' for which a is always enabled but never in the ample set, contradicting **C3**.

If η does not pass through an accepting state infinitely often, there is some $i \geq 0$ such that for all $j \geq i$, $\mathsf{first}(\theta_j)$ is not accepting. But then $(d_j)_{j \geq i}$ is a nondecreasing sequence of positive integers which strictly decreases infinitely often, a contradiction.

4.2 Ample Sets for a Parallel Composition of LTSs

We now describe the specific method used by McRERS to select ample sets. Since this method is similar to existing approaches, such as [26, Algorithm 4.3], we just outline the main ideas.

Let $n \geq 1$, $P = \{1, \ldots, n\}$, and let $M_1, \ldots, M_n$ be LTSs over Act. Write $M_i = (Q_i, A_i, \rightarrow_i, q_i^0)$ and

$$M = M_1 \parallel \cdots \parallel M_n = (Q, A, \rightarrow, q^0).$$

For $a \in A$, let $\mathsf{procs}(a) = \{i \in P \mid a \in A_i\}$. It can be shown that if a and b are dependent actions, then $\mathsf{procs}(a) \cap \mathsf{procs}(b) \neq \emptyset$.

Let $q = (q_1, \ldots, q_n) \in Q$ and $E_i = \mathsf{enabled}(M_i, q_i)$ for $i \in P$. Let

$$R_q = \{(i, j) \in P \times P \mid E_i \cap A_j \neq \emptyset\}.$$

Suppose $C \subseteq P$ is closed under R_q, i.e., for all $i \in C$ and $j \in P$, $(i, j) \in R_q \Rightarrow j \in C$. This implies that if $a \in E_i$ for some $i \in C$ then $\mathsf{procs}(a) \subseteq C$. Define

$$\mathsf{enabled}(C, q) = \mathsf{enabled}(M, q) \cap \bigcup_{i \in C} A_i.$$

Let $E = \mathsf{enabled}(C, q)$. Note $E \subseteq \bigcup_{i \in C} E_i$. Hence for any $a \in E$, $\mathsf{procs}(a) \subseteq C$.

Lemma 3. *On any trace in M starting from q, no action outside of E but dependent on an action in E can occur without an action in E occurring first.*

Proof. Let ζ be a trace in M starting from q, such that no element of E occurs in ζ. We claim no action involving C (i.e., an action a for which $\mathsf{procs}(a) \cap C \neq \emptyset$) can occur in ζ. Otherwise, let x be the first such action. Then $x \in E_i$, for some $i \in C$, so $\mathsf{procs}(x) \subseteq C$. As $x \notin E$, $x \notin \mathsf{enabled}(M, q)$. So some earlier action y in ζ caused x to become enabled, and therefore $\mathsf{procs}(x) \cap \mathsf{procs}(y) \neq \emptyset$, hence $\mathsf{procs}(y) \cap C \neq \emptyset$, contradicting the assumption that x was the first action involving C in ζ.

Now any action b dependent on an action $a \in E$ must satisfy $\mathsf{procs}(a) \cap \mathsf{procs}(b)$ is nonempty. Since $\mathsf{procs}(a) \subseteq C$, $\mathsf{procs}(b) \cap C$ is nonempty. Hence no action dependent on an action in E can occur in ζ. □

We now describe how to find an ample set in the context of NDFS. Let (q, s) be a new product state that has just been pushed onto the outer DFS stack. The relation R_q defined above gives P the structure of a directed graph. Suppose that graph has a strongly connected component C_0 such that all of the following hold for $E = \mathsf{enabled}(C_0, q)$:

1. $E \neq \emptyset$,
2. $E \cap V = \emptyset$,
3. $\mathsf{enabled}(C', q) = \emptyset$ for all SCCs C' reachable from C_0 other than C_0, and
4. E does not contain a "back edge", i.e., if $(q, s) \xrightarrow{a} \sigma$ for some $a \in E$ and $\sigma \in Q \times S$, then σ is not on the outer DFS stack.

Then set $\mathsf{amp}(q, s) = E$. If no such SCC exists, set $\mathsf{amp}(q, s) = \mathsf{enabled}(M, q)$. It follows that **C0**–**C4** hold. Note that the union C of all SCCs reachable from C_0 is closed under R_q, and $\mathsf{enabled}(C, q) = E$, so Lemma 3 guarantees **C1**. For **C3**, we actually have the stronger condition that in any cycle in the reduced space, at least one state is fully enabled. In our implementation, the SCCs are computed using Tarjan's algorithm. Among all SCCs C_0 satisfying the conditions above, we choose one for which $|\mathsf{enabled}(C_0, q)|$ is minimal.

One known issue when combining NDFS with on-the-fly POR is that the inner DFS must explore the same subspace as the outer DFS, i.e., amp must be a deterministic function of its input (q, s) [13]. To accomplish this, McRERS stores one additional integer j in the state: j is the root node of the SCC C_0, or -1 if the state is fully enabled. The outer search saves j in the state, and the inner search uses j to reconstruct the SCC C_0 and the ample set E.

5 Related Work

There has been significant earlier research on the use of partial order reduction to model check LTSs (or the closely related concept of process algebras); see, e.g., [9,11,24–27,29]. To understand how this previous work relates to this paper, we must explain a subtle, but important, distinction concerning how a property is specified. In much of this literature, a property of an LTS with alphabet A is essentially a pair $\pi = (V, T)$, where $V \subseteq A$ is a set of visible actions and T is a set of (finite and infinite) words over V. A property in this sense specifies acceptable behaviors *after invisible actions have been removed.* (See, e.g., Def. 2.4 and preceding comments in [26].) We can translate π to a property P in our sense by taking its inverse image under the projection map:

$$P = \{\zeta \in A^\omega \mid \zeta|_V \in T\}.$$

Note that *P is V-interruptible by definition.* Hence the need to distinguish interruptible properties does not arise in this context.

Much of the earlier work on POR for LTSs deals with the "offline" case, i.e., the construction of a subspace of M that preserves certain classes of properties. In contrast, Theorem 3 deals with an on-the-fly algorithm, i.e., the construction of a subspace of $M \parallel B$. The on-the-fly approach is an essential optimization in model checking, but recent work in the state-based formalism has shown that offline POR schemes do not always generalize easily to on-the-fly algorithms [22].

One earlier work that does describe an on-the-fly model checking algorithm for LTSs is [26] (see also [12], which deals with the same ideas in a state formalism). The property is specified by a *tester process* B. Consistent with the notion of *property* described above, the alphabet of B does not include the invisible actions. Hence, in the parallel composition $M \parallel B$, the tester does not move when M executes an invisible action. In order to specify both finite and infinite words of visible actions, the tester has two kinds of accepting states: "livelock monitor states" and "infinite trace monitor states." (Two additional classes of states for detecting other kinds of violations are not relevant to the discussion here.) A version of the stubborn set theory is used to define the reduced space, and a special condition is used to solve the "ignoring problem" (instead of our **C3**). It would be interesting to compare this algorithm with the one described here.

There are many algorithms for reducing or even minimizing the size of an LTS while preserving various properties, e.g., *bisimulation equivalence* [5] or *divergence preserving bisimilarity* [4]. These algorithms could be applied to the individual components of a parallel composition (taking all visible and communication actions to be "visible"), as a preprocessing step before beginning the model checking search. An exploration of these algorithms, and how they impact POR, is beyond the scope of this paper, but we hope to explore that avenue in future work.

The RERS Challenge [6, 14–16] is an annual event involving a number of different categories of large model checking problems. The "parallel LTL category," offered from 2016 on, is directly relevant to this paper. Each problem in that category consists of a Graphviz "dot" file specifying an LTS as a parallel composition, and a text file containing 20 LTL formulas. The goal is to identify the formulas satisfied by the LTS. The solutions are initially known only to the organizers, and are published after the event. The RERS semantics for LTSs, LTL, and satisfiability are exactly the same as in this paper.

The methods for generating the LTS and the properties are complicated, and have varied over the years, but are designed to satisfy certain hardness guarantees. The approach described in [23] is "... based on the weak refinement ... of convergent systems which preserves an interesting class of temporal properties." It can be seen that the properties preserved by weak refinement are exactly the interruptible properties. While [23] does not describe a method for determining whether a property is interruptible, the authors have informed us that they developed a sufficient condition for an LTL formula to be interruptible, and used this in combination with a random method to generate the formulas for 2016 and 2019. Our analysis (Section 6) confirms that all formulas from 2016 and 2019 are interruptible, while 2017 and 2018 contain some non-interruptible formulas.

There is a well-known way to translate a system and property expressed in an action-based formalism to a state-based formalism. The idea is to add a shared variable *last* which records the last action executed. An LTL formula over actions can be transformed to one over states by replacing each action a with the predicate $last = a$. This is the approach taken in the Promela representations of the parallel problems provided with the RERS challenges.

This translation is semantics-preserving but performance-destroying. Every transition writes to the shared variable *last*, so any state-based POR scheme will assume that no two transitions commute. Furthermore, since the property references *last*, all transitions are visible. This effectively disables POR, even when the property is stutter-invariant, as can be seen in the poor performance of Spin on the RERS Promela models (Section 6). It is possible that there are more effective Spin translations; [28, §2.2], for example, suggests not updating *last* on invisible actions, and adding a global boolean variable that is flipped on every visible action (in addition to updating *last*). We note that this would also require modifying the LTL formula, or specifying the property in some other way. In any case, it suggests another interesting avenue for future work.

6 Experimental Results and Conclusions

We implemented a model checker named McRERS based on the algorithms described in this paper. McRERS is a library and set of command line tools. It is written in sequential C and uses the Spot library [3] for several tasks: (1) determining equivalence of LTL formulas, (2) determining language equivalence of BAs, and (3) converting an LTL formula to a BA. The source code for McRERS as well as all artifacts related to the experiments discussed in this section, are available at `https://vsl.cis.udel.edu/cav2020`. The experiments were run on an 8-core 3.7GHz Intel Xeon W-2145 Linux machine with 256 GB RAM, though McRERS is a sequential program and most experiments required much less memory.

As described in Section 5, each edition of RERS includes a number of problems, each of which comes with 20 LTL formulas. The numbers of problems for years 2016–2019 are, in order, 20, 15, 3, and 9, for a total of 47 problems, or $47 * 20 = 940$ distinct model checking tasks. (Some formulas become identical after renaming propositions.) We used the McRERS *property analyzer* to analyze these formulas to determine which are interruptible; the algorithm used is based on Theorem 1. The results show that all formulas from 2016 and 2019 are interruptible, which agrees with the expectations of the RERS organizers. In 2017, 22 of the 300 formulas are not interruptible; these include

– $\mathbf{GF}\neg\texttt{a111\_SIGTRAP}$,
– $\mathbf{G}[\texttt{a71\_SIGVTALRM} \rightarrow \mathbf{X}\neg\texttt{a71\_SIGVTALRM}]$, and
– $\mathbf{G}[(\texttt{a59\_SIGUSR1} \wedge \mathbf{X}[(\neg\texttt{a112\_SIGHUP})\mathbf{U}\texttt{a59\_SIGUSR1}]) \rightarrow \mathbf{FG}\texttt{a104\_SIGPIPE}]$.

In 2018, 3 of the 60 formulas are not interruptible. The total runtime for the analysis of all 940 formulas was 6 seconds.

We next used the McRERS *automaton analyzer* to create BAs from each of the interruptible formulas, and then to determine which of these Spot-generated BAs was not in interrupt normal form. This uses a straightforward algorithm that iterates over all states and checks the conditions of Def. 11. For each BA not in normal form, the analyzer transforms it to normal form using function norm of Section 3.3. Interestingly, all of the Spot-generated BAs in 2016 and 2019 were already in normal form. Four of the BAs from interruptible formulas in 2017 were not in normal form; all of these formulas had the form $\mathbf{F}[a \vee ((\neg b)\mathbf{W}c)]$. In 2018, 6 interruptible formulas have non-normal BAs; these formulas have several different non-isomorphic forms, some of which are quite complex. The details can be seen on the online archive. The total runtime for this analysis (including writing all BAs to a file) was 11 seconds.

The McRERS model checker parses RERS "dot" and property files to construct an internal representation of a parallel composition $M = M_1 \parallel \cdots \parallel M_n$ of LTSs and a list of LTL formulas. Each formula f is converted to a BA B; if f is interruptible and B is not already in normal form, B is transformed to normal form. The NDFS algorithm is used to determine language emptiness, and if f is interruptible, the POR scheme described in Section 4 is also used. States are saved in a hash table.

One other simple optimization is used regardless of whether f is interruptible. Let αM denote the set of actions labeling at least one transition in M, and define αB similarly. If $\alpha M \neq \alpha B$, then all transitions labeled by an action in $(\alpha M \setminus \alpha B) \cup (\alpha B \setminus \alpha M)$ are removed from the M_i and B; all unreachable states and transitions in the M_i and B are also removed. This is repeated until $\alpha M = \alpha B$.

We applied the model checker to all problems in the 2019 benchmarks. Interestingly, all 180 tasks completed, with the correct results, using at most 8 GB RAM; the times are given in Figure 2.

We also ran these problems with POR turned off, to measure the impact of that optimization. As is often the case with POR schemes, the difference is dramatic. The non-POR tests ran out of memory on our 256 GB machine after problem 106. We show the resources consumed for a representative task in Figure 3; this property holds, so a complete search is required. In terms of number of states or time, the performance differs by about 5 orders of magnitude.

Problem	101	102	103	104	105	106	107	108	109
Components	8	10	12	15	20	25	50	60	70
Time (s)	1	1	1	1	1	1	14	54	432

Fig. 2. Time to solve RERS 2019 parallel LTL problems using McRERS. Each problem comprises 20 LTL formulas. Memory limited to 8 GB. Rows: problem number, number of components in the LTS, and total McRERS wall time rounded up to nearest second.

POR?	States saved	Transitions	Memory (MB)	Time (s)
YES	1.55×10^4	1.55×10^4	1.26×10^2	< 0.1
NO	1.89×10^9	1.35×10^{10}	2.61×10^5	7865.0

Fig. 3. Performance impact of POR on solving RERS 2019 problem 106, formula 1, $(\mathtt{a6} \rightarrow \mathbf{F}\mathtt{a7})\mathbf{W}(\mathtt{a7} \vee \mathtt{a88})$.

As explained in Section 5, the RERS-provided Spin models can not be expected to perform well. We ran the latest version of Spin on these using `-DCOLLAPSE` compression. We show the result for just the first task in Figure 4. There is at least a 4 order of magnitude performance difference (measured in states or time) between the tools. An examination of Spin's output in verbose mode reveals the problem to be as described in Section 5: the full set of enabled transitions is explored at each transition due to the update of the shared variable.

Tool	States	Transitions	Memory(MB)	Time(s)
Spin	8.16×10^7	2.01×10^8	1.09×10^4	292.0
McRERS	1.80×10^2	1.93×10^2	5.06×10^1	< 0.1

Fig. 4. Performance of Spin v6.5.1 and McRERS on RERS 2019 problem 101, property 1. Both tools used POR. Spin used `-DCOLLAPSE` for state compression and `-m100000000` for search depth bound.

The 2016 RERS problems are more challenging for McRERS. The problems are numbered from 101 to 120. To scale beyond problem 111, with a memory bound of 256 GB, additional reduction techniques, such as the component minimization methods discussed in 5, must be used. We plan to carry out a thorough study of those methods and how they interact with POR.

Acknowledgements

We are grateful to Marc Jasper of TU Dortmund for answering many of our questions about the RERS benchmarks, and for coining the term "interruptible" to describe the class of properties that are the topic of this paper.

References

1. Clarke, Jr., E.M., Grumberg, O., Kroening, D., Peled, D., Veith, H.: Model Checking. MIT press, Cambridge, MA, USA, 2 edn. (2018), https://mitpress.mit.edu/books/model-checking-second-edition
2. Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory-efficient algorithms for the verification of temporal properties. Formal Methods in System Design **1**(2), 275–288 (Oct 1992). https://doi.org/10.1007/BF00121128
3. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0 — a framework for LTL and ω-automata manipulation. In: Automated Technology for Verification and Analysis '16. LNCS, vol. 9938, pp. 122–129. Springer (Oct 2016). https://doi.org/10.1007/978-3-319-46520-3_8
4. Eloranta, J., Tienari, M., Valmari, A.: Essential transitions to bisimulation equivalences. Theor. Comput. Sci. **179**(1–2), 397–419 (Jun 1997). https://doi.org/10.1016/S0304-3975(96)00281-2
5. Fernandez, J.C.: An implementation of an efficient algorithm for bisimulation equivalence. Science of Computer Programming **13**(2), 219–236 (1990). https://doi.org/10.1016/0167-6423(90)90071-K
6. Geske, M., Jasper, M., Steffen, B., Howar, F., Schordan, M., van de Pol, J.: RERS 2016: Parallel and sequential benchmarks with focus on LTL verification. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9953, pp. 787–803. Springer (2016). https://doi.org/10.1007/978-3-319-47169-3_59
7. Giannakopoulou, D.: Model checking for concurrent software architectures. Ph.D. thesis, Imperial College of Science, Technology and Medicine, University of London (1999), https://pdfs.semanticscholar.org/0215/b74b21112520569f6e6b930312e228c90e0b.pdf
8. Giannakopoulou, D., Magee, J.: Fluent model checking for event-based systems. In: Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 257–266. ESEC/FSE-11, Association for Computing Machinery, New York, NY, USA (2003). https://doi.org/10.1145/940071.940106
9. Gibson-Robinson, T., Hansen, H., Roscoe, A.W., Wang, X.: Practical partial order reduction for CSP. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NASA Formal Methods. LNCS, vol. 9058, pp. 188–203. Springer (2015). https://doi.org/10.1007/978-3-319-17524-9_14
10. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem, LNCS, vol. 1032. Springer (1996). https://doi.org/10.1007/3-540-60761-7
11. Groote, J.F., Mathijssen, A., Reniers, M., Usenko, Y., van Weerdenburg, M.: The formal specification language mCRL2. In: Brinksma, E., Harel, D., Mader, A., Stevens, P., Wieringa, R. (eds.) Methods for Modelling Software Systems (MMOSS). No. 06351 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany (2007), http://drops.dagstuhl.de/opus/volltexte/2007/862
12. Hansen, H., Penczek, W., Valmari, A.: Stuttering-insensitive automata for on-the-fly detection of livelock properties. Electronic Notes in Theoretical Computer Science **66**(2), 178–193 (2002). https://doi.org/10.1016/S1571-0661(04)80411-0, FMICS'02, 7th International ERCIM Workshop in Formal Methods for Industrial Critical Systems (ICALP 2002 Satellite Workshop)

13. Holzmann, G., Peled, D., Yannakakis, M.: On nested depth first search. In: The Spin Verification System, DIMACS - Series in Discrete Mathematics and Theoretical Computer Science, vol. 32, pp. 23–31. AMS and DIMACS (1997), `https://bookstore.ams.org/dimacs-32/`
14. Jasper, M., Fecke, M., Steffen, B., Schordan, M., Meijer, J., Pol, J.v.d., Howar, F., Siegel, S.F.: The RERS 2017 challenge and workshop (invited paper). In: SPIN 2017. pp. 11–20. ACM (2017). https://doi.org/10.1145/3092282.3098206
15. Jasper, M., Mues, M., Murtovi, A., Schlüter, M., Howar, F., Steffen, B., Schordan, M., Hendriks, D., Schiffelers, R., Kuppens, H., Vaandrager, F.W.: RERS 2019: Combining synthesis with real-world models. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) TACAS 2019. LNCS, vol. 11429, pp. 101–115. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_7
16. Jasper, M., Mues, M., Schlüter, M., Steffen, B., Howar, F.: RERS 2018: CTL, LTL, and reachability. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11245, pp. 433–447. Springer (2018). https://doi.org/10.1007/978-3-030-03421-4_27
17. Michaud, T., Duret-Lutz, A.: Practical stutter-invariance checks for ω-regular languages. In: Fischer, B., Geldenhuys, J. (eds.) SPIN. LNCS, vol. 9232, pp. 84–101. Springer (2015). https://doi.org/10.1007/978-3-319-23404-5_7
18. Peled, D.: Combining partial order reductions with on-the-fly model-checking. Formal Methods in System Design **8**(1), 39–64 (Jan 1996). https://doi.org/10.1007/BF00121262
19. Peled, D., Wilke, T.: Stutter-invariant temporal properties are expressible without the next-time operator. Information Processing Letters **63**(5), 243–246 (Sep 1997). https://doi.org/10.1016/S0020-0190(97)00133-6
20. Peled, D.A.: All from one, one for all: on model checking using representatives. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 409–423 (1993). https://doi.org/10.1007/3-540-56922-7_34
21. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science. pp. 46–57. SFCS '77, IEEE Computer Society (1977). https://doi.org/10.1109/SFCS.1977.32
22. Siegel, S.F.: What's wrong with on-the-fly partial order reduction. In: Dillig, I., Tasiran, S. (eds.) CAV 2019, Part II. LNCS, vol. 11562, pp. 478–495. Springer (2019). https://doi.org/10.1007/978-3-030-25543-5_27
23. Steffen, B., Jasper, M.: Property-preserving parallel decomposition. In: Aceto, L., Bacci, G., Bacci, G., Ingólfsdóttir, A., Legay, A., Mardare, R. (eds.) Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday. LNCS, vol. 10460, pp. 125–145. Springer (2017). https://doi.org/10.1007/978-3-319-63121-9_7
24. Sun, J., Liu, Y., Dong, J.S.: Model checking CSP revisited: Introducing a process analysis toolkit. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. Communications in Computer and Information Science, vol. 17, pp. 307–322. Springer (2008). https://doi.org/10.1007/978-3-540-88479-8_22
25. Valmari, A.: Stubborn sets for reduced state space generation. In: Rozenberg, G. (ed.) Advances in Petri Nets 1990. LNCS, vol. 483, pp. 491–515. Springer (1991). https://doi.org/10.1007/3-540-53863-1_36
26. Valmari, A.: On-the-fly verification with stubborn sets. In: Courcoubetis, C. (ed.) Computer Aided Verification. LNCS, vol. 697, pp. 397–408. Springer (1993). https://doi.org/10.1007/3-540-56922-7_33
27. Valmari, A.: Stubborn set methods for process algebras. In: Proceedings of the DIMACS Workshop on Partial Order Methods in Verification. pp. 213–231. POMIV

'96, American Math. Soc., New York (1997), http://dl.acm.org/citation.cfm?id=266557.266608
28. Valmari, A.: The state explosion problem. In: Reisig, W., Rozenberg, G. (eds.) Advances in Petri Nets: Lectures on Petri Nets I: Basic Models. LNCS, vol. 1491, pp. 429–528. Springer (1998). https://doi.org/10.1007/3-540-65306-6_21
29. Valmari, A.: More stubborn set methods for process algebras. In: Concurrency, Security, and Puzzles: Essays Dedicated to Andrew William Roscoe on the Occasion of His 60th Birthday, LNCS, vol. 10160, pp. 246–271. Springer (2017). https://doi.org/10.1007/978-3-319-51046-0_13
30. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency: Structure versus Automata, LNCS, vol. 1043, pp. 238–266. Springer (1996). https://doi.org/10.1007/3-540-60915-6_6
31. Vardi, M.Y.: Automata-theoretic model checking revisited. In: Cook, B., Podelski, A. (eds.) Verification, Model Checking, and Abstract Interpretation. LNCS, vol. 4349, pp. 137–150. Springer (2007). https://doi.org/10.1007/978-3-540-69738-1_10

A Additional Properties of Interruptible LTL Formulas

A.1 Closure of Intrpt Under Action Equivalence

In this section, we prove

Proposition 8. *If* $f, g \in \mathsf{Form}$ *and* $f \equiv_{\mathsf{A}} g$*, then* $f \in \mathsf{Intrpt} \;\Leftrightarrow\; g \in \mathsf{Intrpt}$.

The statement is easy to prove if $\alpha f = \alpha g$; this is Proposition 4(5). But Proposition 8 is more general because we are not assuming the formulas have the same alphabet. In this case the proof is more subtle.

Definition 13. Let $f \in \mathsf{Form}$ and $\zeta, \eta \in \mathsf{Act}^\omega$. We say ζ and η are *observably equivalent for* f if $\zeta|_{\alpha f} = \eta|_{\alpha f}$.

Given $\zeta \in \mathsf{Act}^\omega$, and $x, y \in \mathsf{Act}$, we write $\zeta[x/y]$ for the sequence obtained by replacing every occurrence of y in ζ by x.

Definition 14. Let $f \in \mathsf{Form}$ and $a \in \mathsf{Act}$. We say a is *irrelevant* to f if the following holds for every $x \in \mathsf{Act} \setminus \alpha f$:

$$\forall \zeta \in \mathsf{Act}^\omega \,.\, (\zeta \models f \;\Leftrightarrow\; \zeta[x/a] \models f). \tag{2}$$

Definition 14 requires that (2) hold *for all* x not in the alphabet of f. Lemma 4 shows that it is enough to check this for one such x:

Lemma 4. *Let* $f \in \mathsf{Form}$ *and* $a \in \mathsf{Act}$. *Suppose* $x \in \mathsf{Act} \setminus \alpha f$ *and* x *satisfies* (2). *Then* a *is irrelevant to* f.

Proof. It suffices to show

$$\forall \zeta \in \mathsf{Act}^\omega,\; x, y \in \mathsf{Act} \setminus \alpha f \,.\, (\zeta[x/a] \models f \;\Leftrightarrow\; \zeta[y/a] \models f). \tag{3}$$

We prove (3) by induction on the structure of f.

It clearly holds if f is *true*. Suppose $f = b \in \mathsf{Act}$. If $a = b$ then neither $\zeta[x/b]$ nor $\zeta[y/b]$ satisfy b, so (3) holds. If $a \neq b$ then

$$\zeta[x/a] \models b \;\Leftrightarrow\; \zeta(0) = b \;\Leftrightarrow\; \zeta[y/a] \models b,$$

so (3) holds again.

Suppose (3) holds for f, and x is not in

$$\begin{aligned} \zeta[x/a] \models \neg f \;&\Leftrightarrow\; \zeta[x/a] \not\models f \\ &\Leftrightarrow\; \zeta[y/a] \not\models f \\ &\Leftrightarrow\; \zeta[y/a] \models \neg f \end{aligned}$$

showing (3) holds for $\neg f$.

Suppose (3) holds for f and g, and x, y are not in $\alpha(f \wedge g) = \alpha(f) \cup \alpha(g)$. Then

$$\begin{aligned}\zeta[x/a] \models f \wedge g &\Leftrightarrow \zeta[x/a] \models f \text{ and } \zeta[x/a] \models g \\ &\Leftrightarrow \zeta[y/a] \models f \text{ and } \zeta[y/a] \models g \\ &\Leftrightarrow \zeta[y/a] \models f \wedge g,\end{aligned}$$

proving (3) for $f \wedge g$. Moreover,

$$\begin{aligned}\zeta[x/a] \models f\mathbf{U}g &\Leftrightarrow \exists i.(i \geq 0 \wedge \zeta[x/a]^i \models g \wedge \forall j.(0 \leq j < i \rightarrow \zeta[x/a]^j \models f)) \\ &\Leftrightarrow \exists i.(i \geq 0 \wedge \zeta^i[x/a] \models g \wedge \forall j.(0 \leq j < i \rightarrow \zeta^j[x/a] \models f)) \\ &\Leftrightarrow \exists i.(i \geq 0 \wedge \zeta^i[y/a] \models g \wedge \forall j.(0 \leq j < i \rightarrow \zeta^j[y/a] \models f)) \\ &\Leftrightarrow \exists i.(i \geq 0 \wedge \zeta[y/a]^i \models g \wedge \forall j.(0 \leq j < i \rightarrow \zeta[y/a]^j \models f)) \\ &\Leftrightarrow \zeta[y/a] \models f\mathbf{U}g,\end{aligned}$$

proving (3) for $f\mathbf{U}g$. □

Lemma 5 states that any action not in the alphabet of f is irrelevant to f:

Lemma 5. *Let $f \in \mathsf{Form}$ and $a \in \mathsf{Act} \setminus \alpha f$. Then a is irrelevant to f.*

Proof. Let $x = a$. Then $\zeta[x/a] = \zeta$, so (2) clearly holds. By Lemma 4, a is irrelevant to f. □

Actions that do occur in f may also be irrelevant to f. For example, $a \wedge \neg a$ is equivalent to *false*, so clearly a is irrelevant to $a \wedge \neg a$. Similarly, if a and b are distinct actions, then $a \wedge b$ is action-equivalent to *false*, so both a and b are irrelevant to $a \wedge b$. More generally:

Lemma 6. *Suppose $f, g \in \mathsf{Form}$, $f \equiv_{\mathsf{A}} g$, and $a \in \mathsf{Act}$. Then a is irrelevant to f if, and only if, a is irrelevant to g.*

Proof. Suppose a is irrelevant to f. Since Act is infinite, there is some $x \in \mathsf{Act} \setminus (\alpha f \cup \alpha g)$. Then for any $\zeta \in \mathsf{Act}^\omega$,

$$\zeta \models g \Leftrightarrow \zeta \models f \Leftrightarrow \zeta[x/a] \models f \Leftrightarrow \zeta[x/a] \models g.$$

By Lemma 4, a is irrelevant to g. □

Now we prove Proposition 8:

Proof. Let $Y = \alpha f \setminus \alpha g$. Say $Y = \{y_1, \ldots, y_n\}$. Let $x \in \mathsf{Act} \setminus (\alpha f \cup \alpha g)$. For $\zeta \in \mathsf{Act}^\omega$, we write $\zeta[x/Y]$ as shorthand for $\zeta[x/y_1] \cdots [x/y_n]$. Note

$$\zeta[x/Y]|_{\alpha f} = \zeta|_{\alpha f \cap \alpha g} = \zeta|_{\alpha g}|_{\alpha f \cap \alpha g}. \tag{4}$$

Suppose $f \in \mathsf{Intrpt}$. We must show $g \in \mathsf{Intrpt}$. So assume ζ, $\eta \in \mathsf{Act}^\omega$ and $\zeta|_{\alpha g} = \eta|_{\alpha g}$. By (4),

$$\zeta[x/Y]|_{\alpha f} = \zeta|_{\alpha g}|_{\alpha f \cap \alpha g} = \eta|_{\alpha g}|_{\alpha f \cap \alpha g} = \eta[x/Y]|_{\alpha f},$$

whence $\zeta[x/Y]$ and $\eta[x/Y]$ are observably equivalent for f.

By Lemma 5, every action in Y is irrelevant to g. Therefore

$$\begin{aligned}\zeta \models g &\Leftrightarrow \zeta[x/Y] \models g && \text{by Definition 14}\\ &\Leftrightarrow \zeta[x/Y] \models f && \text{since } f \equiv_A g\\ &\Leftrightarrow \eta[x/Y] \models f && \text{as } \zeta[x/Y], \eta[x/Y] \text{ are observably equivalent for } f\\ &\Leftrightarrow \eta[x/Y] \models g && \text{since } f \equiv_A g\\ &\Leftrightarrow \eta \models g && \text{by Definition 14.}\end{aligned}$$

Hence $g \in \mathsf{Intrpt}$. □

A.2 Closure of Intrpt Under All LTL Operators Other Than X

In this section, we prove

Proposition 9. *Suppose $f, g \in \mathsf{Intrpt}$. Then true, $\neg f$, $f \wedge g$, $f \vee g$, $f \rightarrow g$, $\mathbf{F}f$, $\mathbf{G}f$, and $f\mathbf{U}g$ are all in Intrpt.*

Proof. The proof is by induction over the formula structure.

Suppose ζ and η are observably equivalent for f. Then

$$\zeta \models \neg f \;\Leftrightarrow\; \zeta \not\models f \;\Leftrightarrow\; \eta \not\models f \;\Leftrightarrow\; \eta \models \neg f,$$

so $\neg f \in \mathsf{Intrpt}$.

Suppose ζ and η are observably equivalent for $f \wedge g$. Note $\alpha(f \wedge g) = \alpha f \cup \alpha g$, so

$$\zeta|_{\alpha f} = \zeta|_{\alpha(f\wedge g)}|_{\alpha f} = \eta|_{\alpha(f\wedge g)}|_{\alpha f} = \eta|_{\alpha f},$$

so ζ and η are observably equivalent for f. Similarly, ζ and η are observably equivalent for g. Hence

$$\begin{aligned}\zeta \models f \wedge g &\Leftrightarrow \zeta \models f \text{ and } \zeta \models g\\ &\Leftrightarrow \eta \models f \text{ and } \eta \models g\\ &\Leftrightarrow \eta \models f \wedge g,\end{aligned}$$

which shows $f \wedge g \in \mathsf{Intrpt}$.

Let us show the case for $f\mathbf{U}g$. The other cases follow since they are action-equivalent to formulas expressed using $\mathbf{U}$, *true*, $\neg$, and $\wedge$.

Suppose $\zeta \models f\mathbf{U}g$. Then there is some $i \geq 0$ such that $\zeta^i \models g$ and $\zeta^j \models f$ whenever $0 \leq j < i$.

Since ζ and η are observably equivalent, there exists $i' \geq 0$ such that the prefix of length i of ζ is observably equivalent to the prefix of length i' of η, and the suffix ζ^i is observably equivalent to $\eta^{i'}$.

For $j' < i'$, $\eta^{j'} \models f$ since $\eta^{j'}$ is observably equivalent to ζ^j for some j satisfying $0 \leq j < i$. Similarly, $\eta^{i'} \models g$ since $\eta^{i'}$ is observably equivalent to ζ^i. Hence $\eta \models f\mathbf{U}g$, as required. □

A.3 Proof of Proposition 5

In this section, we prove

Proposition 5. *There exist* $\mathsf{Pos}, \mathsf{Neg} \subseteq \mathsf{Form}$ *such that (i) for all* $f, f' \in \mathsf{Form}$,

$$(f \in \mathsf{Pos} \wedge f' \equiv_{\mathsf{A}} f) \Rightarrow f' \in \mathsf{Pos}$$
$$(f \in \mathsf{Neg} \wedge f' \equiv_{\mathsf{A}} f) \Rightarrow f' \in \mathsf{Neg},$$

and (ii) for all $a \in \mathsf{Act}$, $f_1, f_2 \in \mathsf{Intrpt}$, $g_1, g_2 \in \mathsf{Pos}$, *and* $h_1, h_2 \in \mathsf{Neg}$,

$$\begin{aligned} \mathit{false},\ a,\ \neg h_1,\ g_1 \wedge g_2,\ g_1 \vee g_2,\ a \wedge f_1,\ a \wedge \mathbf{X} f_1 &\in \mathsf{Pos} \\ \mathit{true},\ \neg a,\ \neg g_1,\ h_1 \wedge h_2,\ h_1 \vee h_2,\ \neg a \vee f_1,\ \neg a \vee \mathbf{X} f_1 &\in \mathsf{Neg} \\ \mathit{true},\ \mathit{false},\ f_1 \wedge f_2,\ f_1 \vee f_2,\ \neg f_1,\ \mathbf{F} g_1,\ \mathbf{G} h_1,\ f_1 \mathbf{U} f_2,\ h_1 \mathbf{U} g_1,\ h_1 \mathbf{U} f_1 &\in \mathsf{Intrpt}. \end{aligned}$$

Definition of positive and negative formulas. Let Pos be the set of all $f \in \mathsf{Form}$ satisfying both of the following:

$$\forall a \in \mathsf{Act}, \zeta, \eta \in \mathsf{Act}^\omega \,.\, (\zeta|_{\alpha f} = \eta|_{\alpha f} \Rightarrow (a \circ \zeta \models f \iff a \circ \eta \models f)) \tag{5}$$

$$\forall \zeta \in \mathsf{Act}^\omega \,.\, (\zeta(0) \notin \alpha f \Rightarrow \zeta \not\models f). \tag{6}$$

We say a formula in Pos is *positive*.

Let Neg be the set of all $f \in \mathsf{Form}$ satisfying both (5) and

$$\forall \zeta \in \mathsf{Act}^\omega \,.\, (\zeta(0) \notin \alpha f \Rightarrow \zeta \models f). \tag{7}$$

We say a formula in Neg is *negative*.

Lemma 7. *Suppose* $f, g \in \mathsf{Form}$ *and* $f \equiv_A g$. *All of the following hold:*

1. $f \in \mathsf{Pos} \iff g \in \mathsf{Pos}$,
2. $f \in \mathsf{Neg} \iff g \in \mathsf{Neg}$,
3. $f \in \mathsf{Pos} \iff \neg f \in \mathsf{Neg}$, *and*
4. $f \in \mathsf{Neg} \iff \neg f \in \mathsf{Pos}$.

Proof. Clearly, (5) holds for f if and only if (5) holds for $\neg f$. Equation (6) holds for f if and only if (7) holds for $\neg f$. This proves Lemma 7(3). Moreover, (7) holds for f if and only if (6) holds for $\neg f$, proving Lemma 7(4).

Now if Lemma 7(1) holds, we can prove Lemma 7(2): if f is negative then $\neg f$ is positive, so since $\neg f \equiv_A \neg g$, $\neg g$ is positive, therefore g is negative.

We now turn to the proof of Lemma 7(1). Let $Y = \alpha f \setminus \alpha g$ and $x \in \mathsf{Act} \setminus (\alpha f \cup \alpha g)$. Assume f is positive. We wish to show g is positive.

We first show g satisfies (5). So assume $\zeta|_{\alpha g} = \eta|_{\alpha g}$ and $a \circ \zeta \models g$. We must show $a \circ \eta \models g$. First, since $f \equiv_A g$, we have

$$a \circ \zeta \models f. \tag{8}$$

Moreover,

$$\begin{aligned} \zeta[x/Y]|_{\alpha f} &= \zeta|_{\alpha f \cap \alpha g} = \zeta|_{\alpha g}|_{\alpha f \cap \alpha g} = \eta|_{\alpha g}|_{\alpha f \cap \alpha g} = \eta|_{\alpha f \cap \alpha g} \\ &= \eta[x/Y]|_{\alpha f}. \end{aligned} \tag{9}$$

By Lemma 5, every element of Y is irrelevant to g, and by Lemma 6, every element of Y is irrelevant to f. From (8), we can therefore conclude

$$(a \circ \zeta)[x/Y] \models f. \tag{10}$$

We claim $a \notin Y$. For if $a \in Y$, then $(a \circ \zeta)[x/Y] = x \circ \zeta[x/Y] \models f$ but $(x \circ \zeta[x/Y])(0) = x \notin \alpha f$, so (6) implies $x \circ \zeta[x/Y] \not\models f$, contradicting (10).

Therefore

$$a \circ \zeta[x/Y] = (a \circ \zeta)[x/Y] \models f. \tag{11}$$

But f is positive, so by (5), (9), and (11),

$$a \circ \eta[x/Y] \models f.$$

Since $f \equiv_A g$,

$$(a \circ \eta)[x/Y] = a \circ \eta[x/Y] \models g,$$

and since Y is irrelevant to g, $a \circ \eta \models g$, completing the proof that g satisfies (5).

We next show g satisfies (6). Let $\zeta \in \mathsf{Act}^\omega$, $a = \zeta(0)$, and suppose $a \notin \alpha g$. Say $\zeta = a \circ \xi$. We have

$$\zeta[x/a] = (a \circ \xi)[x/a] = x \circ \xi[x/a].$$

Since $x \notin \alpha f$ and f satisfies (6), $x \circ \xi[x/a] \not\models f$. Since $f \equiv_A g$, $\zeta[x/a] \not\models g$. By Lemma 5, a is irrelevant to g, so $\zeta \not\models g$. This shows g satisfies (6), completing the proof of Lemma 7(1). □

Positive formulas. The proof of Proposition 5 is by induction on formula structure. We begin with the formulas in Pos.

Case false. Since formula *false* trivially satisfies (5) and (6), *false* is clearly positive.

Case a. Let $a \in \mathsf{Act}$. For any $x \in \mathsf{Act}$,

$$x \circ \zeta \models a \iff x = a \iff x \circ \eta \models a,$$

so (5) holds for the formula a. Clearly, if ζ does not start with a then ζ cannot satisfy a, so (6) holds as well. Hence the formula a is positive.

Case $\neg h$. Suppose $h \in \mathsf{Neg}$. By the inductive hypothesis, h is negative. By Lemma 7(4), $\neg h$ is positive.

Case $g_1 \wedge g_2$. Suppose $g_1, g_2 \in \mathsf{Pos}$. By the inductive hypothesis, g_1 and g_2 are positive. We show $g_1 \wedge g_2$ is positive. Let $S = \alpha(g_1 \wedge g_2) = \alpha g_1 \cup \alpha g_2$. Suppose $a \in \mathsf{Act}$, $\zeta, \eta \in \mathsf{Act}^\omega$, $\zeta|_S = \eta|_S$, and $a \circ \zeta \models g_1 \wedge g_2$. Then $a \circ \zeta \models g_1$ and

$$\zeta|_{\alpha g_1} = \zeta|_S|_{\alpha g_1} = \eta|_S|_{\alpha g_1} = \eta|_{\alpha g_1}.$$

Since g_1 is positive, $a \circ \eta \models g_1$. A similar statement holds with g_2 in place of g_1. Hence $a \circ \eta \models g_1 \wedge g_2$, and $g_1 \wedge g_2$ satisfies (5). Furthermore, if $\zeta(0) \notin S$, then in particular $\zeta(0) \notin \alpha g_1$, so $\zeta \not\models g_1$, hence $\zeta \not\models g_1 \wedge g_2$. Therefore $g_1 \wedge g_2$ also satisfies (6), and $g_1 \wedge g_2$ is positive.

Case $g_1 \vee g_2$. One can argue exactly as above to see that $g_1 \vee g_2$ satisfies (5). If $\zeta(0) \notin S$, then $\zeta(0) \notin \alpha g_1$, whence $\zeta \not\models g_1$, and $\zeta(0) \notin \alpha g_2$, whence $\zeta \not\models g_2$. Therefore $\zeta \not\models g_1 \vee g_2$, so $g_1 \vee g_2$ is positive.

Cases $a \wedge f$ *and* $a \wedge \mathbf{X}f$. Suppose $a \in \mathsf{Act}$ and $f \in \mathsf{Intrpt}$. Let $S = \{a\} \cup \alpha f$, and assume $b \in \mathsf{Act}$ and $\zeta|_S = \eta|_S$.

Suppose $b \circ \zeta \models a \wedge f$. Then $a = b$ and $a \circ \zeta \models f$. Moreover,

$$(a \circ \zeta)|_{\alpha f} = (a \circ \zeta)|_S|_{\alpha f} = (a \circ \eta)|_S|_{\alpha f} = (a \circ \eta)|_{\alpha f}. \quad (12)$$

Since $f \in \mathsf{Intrpt}$, $a \circ \eta \models f$, hence $a \circ \eta \models a \wedge f$, and $a \wedge f$ satisfies (5). Suppose $\zeta(0) \notin S$. Then, in particular, $\zeta(0) \neq a$, so $\zeta \not\models a \wedge f$, and $a \wedge f$ satisfies (6). So $a \wedge f$ is positive.

Suppose instead $b \circ \zeta \models a \wedge \mathbf{X}f$. Then $a = b$ and $a \circ \zeta \models \mathbf{X}f$. Hence $\zeta \models f$. Moreover, $\zeta|_{\alpha f} = \eta|_{\alpha f}$. Since $f \in \mathsf{Intrpt}$, $\eta \models f$. Hence $a \circ \eta \models a \wedge \mathbf{X}f$, i.e., $a \wedge \mathbf{X}f$ satisfies (5). The argument that $a \wedge \mathbf{X}f$ satisfies (6) is identical to the one above. So $a \wedge \mathbf{X}f$ is positive.

Negative formulas. Each formula in Neg is easily seen to be equivalent to the negation of a formula in Pos. By Lemma 7, this completes the inductive step for Neg.

Interruptible formulas. Proposition 9 has already established that the interruptible formulas are closed under all LTL operators other than $\mathbf{X}$. Moreover, $\mathbf{F}g \equiv \mathit{true}\mathbf{U}g$, and $\mathit{true} \in \mathsf{Neg}$; and $\mathbf{G}h \equiv \neg\mathbf{F}\neg h$. This leaves two cases: $h\mathbf{U}g$ and $h\mathbf{U}f$.

Case $h\mathbf{U}g$. Suppose h is negative and g is positive. We will show $h\mathbf{U}g \in \mathsf{Intrpt}$. Let $Y = \alpha h \cup \alpha g$. Suppose ζ and η are observably equivalent for $h\mathbf{U}g$ and $\zeta \models h\mathbf{U}g$. Say $\zeta^i \models g$ and $\zeta^j \models h$ for $j < i$.

Let $a = \zeta^i(0) = \zeta(i)$. Since g is positive, $a \in \alpha g$. Now since $\zeta|_Y = \eta|_Y$, there is a unique integer i' such that

$$\begin{aligned} \zeta &= \zeta_1 \circ a \circ \zeta^{i+1} \\ \eta &= \eta_1 \circ a \circ \eta^{i'+1} \\ \zeta_1|_Y &= \eta_1|_Y \\ \zeta^{i+1}|_Y &= \eta^{i'+1}|_Y, \end{aligned}$$

where ζ_1 is the prefix of length i of ζ and η_1 is the prefix of length i' of η. Since $a \circ \zeta^{i+1} = \zeta^i \models g$, we have $a \circ \eta^{i'+1} = \eta^{i'} \models g$.

Suppose $0 \leq j' < i'$. Let $b = \eta(j')$. If $b \notin Y$, then since h is negative, $\eta^{j'} \models h$. If $b \in Y$, then there exists a unique integer j such that

$$\begin{aligned} \zeta &= \zeta_2 \circ b \circ \zeta^{j+1} \\ \eta &= \eta_2 \circ b \circ \eta^{j'+1} \\ \zeta_2|_Y &= \eta_2|_Y \\ \zeta^{j+1}|_Y &= \eta^{j'+1}|_Y, \end{aligned}$$

where ζ_2 is the prefix of length j of ζ and η_2 is the prefix of length j' of η. Furthermore, $j < i$, so $b \circ \zeta^{j+1} \models h$. Since h is negative, $\eta^{j'} = b \circ \eta^{j'+1} \models h$. This shows $\eta \models h\mathbf{U}g$, so $h\mathbf{U}g \in \mathsf{Intrpt}$.

Case $h\mathbf{U}f$. Suppose h is negative and $f \in \mathsf{Intrpt}$. We will show $h\mathbf{U}f \in \mathsf{Intrpt}$. Let $Y = \alpha h \cup \alpha g$. Suppose ζ and η are observably equivalent for $h\mathbf{U}f$ and $\zeta \models h\mathbf{U}f$. Let i be the least nonnegative integer such that $\zeta^i \models f$ and $\zeta^j \models h$ for $j < i$.

If $i = 0$ then $\zeta \models f$, and therefore $\eta \models f$, since $f \in \mathsf{Intrpt}$. Hence $\eta \models h\mathbf{U}f$, as required.

So assume $i > 0$. Let $a = \zeta(i-1)$. We must have $a \in \alpha f$, since otherwise ζ^{i-1} and ζ^i would be observably equivalent for f, and therefore $\zeta^{i-1} \models f$, contradicting the minimality of i.

There exists a unique nonnegative integer i' such that

$$\begin{aligned} \zeta &= \zeta_1 \circ a \circ \zeta^{i} \\ \eta &= \eta_1 \circ a \circ \eta^{i'} \\ \zeta_1|_Y &= \eta_1|_Y \\ \zeta^{i}|_Y &= \eta^{i'}|_Y, \end{aligned}$$

where ζ_1 is the prefix of length i of ζ and η_1 is the prefix of length i' of η.

Since $f \in \mathsf{Intrpt}$, $\eta^{i'} \models f$. Suppose $0 \leq j' < i'$. We will show $\eta^{j'} \models h$. If $j' = i' - 1$, then $\eta^{j'}|_Y = a \circ \eta^{i'} \models h$ since h is negative, $a \circ \zeta^i \models h$, and $\zeta^i|_Y = \eta^i|_Y$.

Suppose $j' < i' - 1$. Then we argue just as in the $h\mathbf{U}g$ case that $\eta^{j'} \models h$. This completes the proof of Proposition 5.

A.4 An Alternative Characterization of Irrelevance

We now turn to an equivalent formulation of the notion of *irrelevant* actions. If $f, g \in \mathsf{Form}$ and $a \in \mathsf{Act}$, we write $f[g/a]$ for the formula obtained by replacing every occurrence of a in f with g.

Lemma 8. *Let $f \in \mathsf{Form}$, $a \in \mathsf{Act}$, $x \in \mathsf{Act} \setminus \alpha f$, and $\zeta \in \mathsf{Act}^\omega$. Then*

$$\zeta[x/a] \models f \;\Leftrightarrow\; \zeta \models f[\mathit{false}/a] \tag{13}$$

Proof. Proof is by induction on the structure of f. The lemma certainly holds for the formula *true*. Suppose $b \in \mathsf{Act}$ and we wish to show it holds for the formula b. If $a \neq b$, then

$$\zeta[x/a] \models b \;\Leftrightarrow\; \zeta(0) = b \;\Leftrightarrow\; \zeta \models b = b[\mathit{false}/a],$$

so (13) holds. If $a = b$, then

$$\zeta[x/a] \models a \;\Leftrightarrow\; \mathit{false} \;\Leftrightarrow\; \zeta \models \mathit{false} = a[\mathit{false}/a],$$

so (13) holds in this case as well.

Suppose the Lemma holds for f and g. Then

$$\begin{aligned} \zeta[x/a] \models f \wedge g \;&\Leftrightarrow\; \zeta[x/a] \models f \text{ and } \zeta[x/a] \models g \\ &\Leftrightarrow\; \zeta \models f[\mathit{false}/a] \text{ and } \zeta \models g[\mathit{false}/a] \\ &\Leftrightarrow\; \zeta \models f[\mathit{false}/a] \wedge g[\mathit{false}/a] \\ &\Leftrightarrow\; \zeta \models (f \wedge g)[\mathit{false}/a], \end{aligned}$$

so the Lemma holds for $f \wedge g$. Furthermore,

$$\begin{aligned} \zeta[x/a] \models \neg f \;&\Leftrightarrow\; \zeta[x/a] \not\models f \\ &\Leftrightarrow\; \zeta \not\models f[\mathit{false}/a] \\ &\Leftrightarrow\; \zeta \models \neg(f[\mathit{false}/a]) \\ &\Leftrightarrow\; \zeta \models (\neg f)[\mathit{false}/a], \end{aligned}$$

so it holds for $\neg f$. Finally,

$$\begin{aligned} \zeta[x/a] \models f\mathbf{U}g \;&\Leftrightarrow\; \exists i.(i \geq 0 \wedge (\zeta[x/a]^i \models g) \wedge \forall j.(0 \leq j < i \rightarrow \zeta[x/a]^j \models f)) \\ &\Leftrightarrow\; \exists i.(i \geq 0 \wedge (\zeta^i[x/a] \models g) \wedge \forall j.(0 \leq j < i \rightarrow \zeta^j[x/a] \models f)) \\ &\Leftrightarrow\; \exists i.(i \geq 0 \wedge (\zeta^i \models g[\mathit{false}/a]) \wedge \forall j.(0 \leq j < i \rightarrow \zeta^j \models f[\mathit{false}/a])) \\ &\Leftrightarrow\; \zeta \models f[\mathit{false}/a]\mathbf{U}g[\mathit{false}/a] \\ &\Leftrightarrow\; \zeta \models (f\mathbf{U}g)[\mathit{false}/a], \end{aligned}$$

so the Lemma holds for $f\mathbf{U}g$. ☐

Proposition 10. *Let $f \in \mathsf{Form}$ and $a \in \mathsf{Act}$. Then a is irrelevant to f if, and only if, $f \equiv_A f[\mathit{false}/a]$.*

Proof. Let x be any element of $\mathsf{Act} \setminus \alpha f$. (Such an element exists since Act is infinite and αf is finite.)

Suppose a is irrelevant to f. Let $\zeta \in \mathsf{Act}^{\omega}$. Then

$$\begin{aligned} \zeta \models f &\Leftrightarrow \zeta[x/a] \models f && \text{by Definition 14} \\ &\Leftrightarrow \zeta \models f[\mathit{false}/a] && \text{by Lemma 8} \end{aligned}$$

whence $f \equiv_A f[\mathit{false}/a]$.

Suppose $f \equiv_A f[\mathit{false}/a]$. Let $\zeta \in \mathsf{Act}^{\omega}$. Then

$$\begin{aligned} \zeta \models f &\Leftrightarrow \zeta \models f[\mathit{false}/a] && \text{by Definition 4} \\ &\Leftrightarrow \zeta[x/a] \models f && \text{by Lemma 8.} \end{aligned}$$

By Lemma 4, a is irrelevant to f. $\square$