FI MU

Faculty of Informatics
Masaryk University Brno

Partial Order Reduction for State/Event LTL

by

N. Beneš
L. Brim
I. Černá
J. Sochor
P. Vařeková
B. Zimmerova

FI MU Report Series FIMU-RS-2008-07

 July 2008

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

Partial Order Reduction for State/Event LTL

N. Beneš[*] L. Brim[*] I. Černá[†] J. Sochor[†] P. Vařeková[†]
B. Zimmerova[†]
Faculty of Informatics, Masaryk University,
Botanická 68a, 60200 Brno,
Czech Republic.
{xbenes3,brim,cerna,sochor,xvareko1,zimmerova}@fi.muni.cz

July 18, 2008

Abstract

Software systems assembled from a large number of autonomous components become an interesting target for formal verification due to the issue of correct interplay in component interaction. State/event LTL [6, 5] incorporates both states and events to express important properties of component-based software systems.

The main contribution of the paper is a partial order reduction technique for verification of state/event LTL properties. The core of the partial order reduction is a novel notion of stuttering equivalence which we call state/event stuttering equivalence. The positive attribute of the equivalence is that it can be resolved with existing methods for partial order reduction. State/event LTL properties are, in general, not preserved under state/event stuttering equivalence. To this end we define a new logic, called weak state/event LTL, which is invariant under the new equivalence.

1 Introduction

Increasing complexity in software development stimulates application of new techniques that help to deliver systems in shorter time and with lower costs. One of such techniques is the component-based development, that builds software systems out of

[*]The author has been supported by grant no. 1ET408050503.
[†]The author has been supported by grant no. 1ET400300504.

prefabricated autonomous components, often developed with no knowledge of their deployment context. Under such conditions, interaction among components in the system becomes a crucial issue in the system correctness.

Verification of component-based systems. Similarly to communicating processes, interaction of components can be formalized in terms of labelled transition systems, representing communicational behaviour of the components, and correctness of the systems in a temporal logic. In practice, real systems are composed of a large number of components which are often independent on each other and run concurrently. In such cases, automated verification becomes challenging due to their size and complexity. This motivates the search of component-specific attributes, which can be exploited in order to make the verification feasible.

Correctness attributes. One of the crucial observations in verification of component interaction in component-based systems is that the correctness attributes often highlight interaction among specific components which form only a small part of the system. Even if the rest of the system is also important as it may coordinate these components, with appropriate reduction techniques a large portion of its complexity could be abstracted away during verification.

Partial-order reduction technique. One of the techniques successfully employed to state-space reduction is the *partial order reduction*. This technique is able to identify redundancies in the model during the verification process, commonly caused by interleaving of independent actions. This allows the technique to omit generation of some of them while at least one representative of each equivalence class remains part of the actually verified model.

State/event temporal logic. In component-based systems, as in any modular programs in general, communication among components proceeds via events, which represent message passing, service calls, delivery of return values, etc. At the same time, components preserve also persistent state information about current values of their attributes. The adequate logic formalizing properties of these systems hence should be able to express both state-based and action-based properties, as well as their combinations. Research conducted on this topic resulted in the state/event LTL [6, 5]. For the logic, however, there is no partial order reduction method known at the time. The situation is complicated for its fragment, the action-based LTL, as well.

Contribution. The main contribution of this paper is a partial order reduction method for state/event LTL, and for action-based LTL as its special case. The whole framework is moreover defined in a way that it can be turned at no additional cost into the standard partial order reduction problem for state-based LTL at the end. Hence, it can be resolved with known and widely implemented techniques.

For the reduction to have required effect one needs to identify an equivalence of two runs. The equivalence should allow for considerable level of reduction, while preserving temporal properties that reflect meaningful correctness attributes for the studied systems. We define an equivalence relation, *state/event stuttering equivalence*, driven by the correctness attributes of component-based systems, highlighting only the interesting interaction of components as discussed above, and characterize the state/event LTL properties preserved by the equivalence in terms of a new logic named *weak state/event LTL*.

Outline of the paper. After a brief review of related work in Section 2, we define basic terms and structures in Section 3. The equivalence and the corresponding logic are introduced in Section 5, which is preceded by thorough motivation underlying their definition in Section 4. Finally, the partial order reduction technique is presented in Section 6, which is followed by conclusion and future directions in Section 7.

2 Related Work

A combination of state-based and action-based linear temporal logic, named state/event LTL, has been studied in [6, 5]. The authors argue that formalisms including both states and actions are suited for modelling of modular systems, including component-based systems, better than pure state-based or action-based approaches. It is also shown that the automata-based verification method for state-based LTL [10] can be modified to a verification method for state/event LTL in a straightforward way and at no additional cost of time and space. As noted by the authors, the results indicate the importance of further research in reduction techniques. The partial order reduction is suggested as a future direction. Its need was also discovered in our recent work on verification of component-based systems [18, 2].

The partial order reduction method was originally introduced in three independent works [17, 14, 11]. The approach has been further developed, but in connection with linear temporal properties, state-based LTL has always been assumed. The reason for leaving action-based and state/event LTL behind is most likely because the correctness of the partial order reduction method is based on the concept of stuttering invariance of properties [16]. To the best of our knowledge, the stuttering concept has currently no convenient analogue for neither state/event nor action-based LTL.

An approach that relates actions and stuttering equivalence is the temporal logic of actions [12], where the formulae are constructed in a way that they are stuttering invariant. However, the *actions* are formulated in terms of changes of state propositions and/or variables, not allowing an arbitrary concept of actions. In our approach, we adopt a more general attitude to actions, as we consider arbitrary actions not tied to the properties of states.

3 Basic Definitions

As a general modelling formalism for state/event systems we use *labelled Kripke structures*. Many automata-based approaches, such as Component-Interaction Automata [2, 18], Interface Automata [9] and I/O automata [13], can be easily translated to labelled Kripke structures.

Definition 3.1 (LKS). *A* labelled Kripke structure (LKS) *is a 6-tuple* $(S, \mathsf{Act}, \Delta, s_{\mathsf{init}}, \mathsf{Ap}, \mathcal{L})$ *where* S *is a nonempty set of* states, Act *is a finite set of* actions, $\Delta \subseteq S \times \mathsf{Act} \times S$ *is a* transition relation, $s_{\mathsf{init}} \in S$ *is an* initial state, Ap *is a finite set of* atomic propositions *and* $\mathcal{L}\colon S \to 2^{\mathsf{Ap}}$ *is a* state-labelling function. *Instead of* $(s, a, s') \in \Delta$, *we also write* $s \xrightarrow{a} s'$.

A run π *of an LKS is an infinite alternating sequence of states and actions* $\pi = s_0, a_0, s_1, a_1, \ldots$ *such that* $\forall i : s_i \xrightarrow{a_i} s_{i+1}$. *We call a run* initial *if* $s_0 = s_{\mathsf{init}}$. *Given a run* π, *we also define: the* i*th subrun of* π *as* $\pi^i = s_i, a_i, s_{i+1}, a_{i+1}, \ldots$, *the* i*th state of* π *as* $\pi(i) = s_i$ *and the* i*th action of* π *as* $\ell(\pi, i) = a_i$.

Other kinds of transition systems can be naturally translated into LKSs. The most commonly used are the Kripke structures, which correspond to LKSs without transition labels (i.e. actions), and the labelled transition systems, which correspond to LKSs without atomic propositions and state labelling.

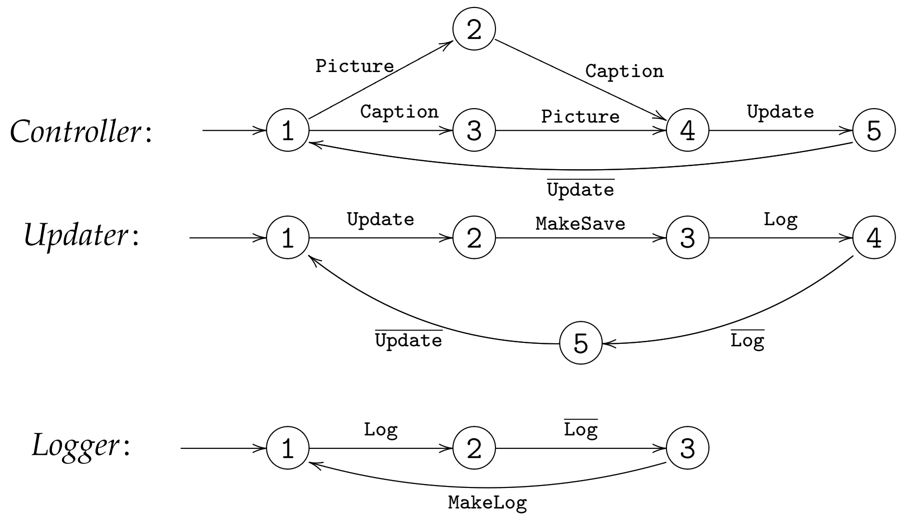

Figure 1: Models of the *Controller*, *Updater* and *Logger* components

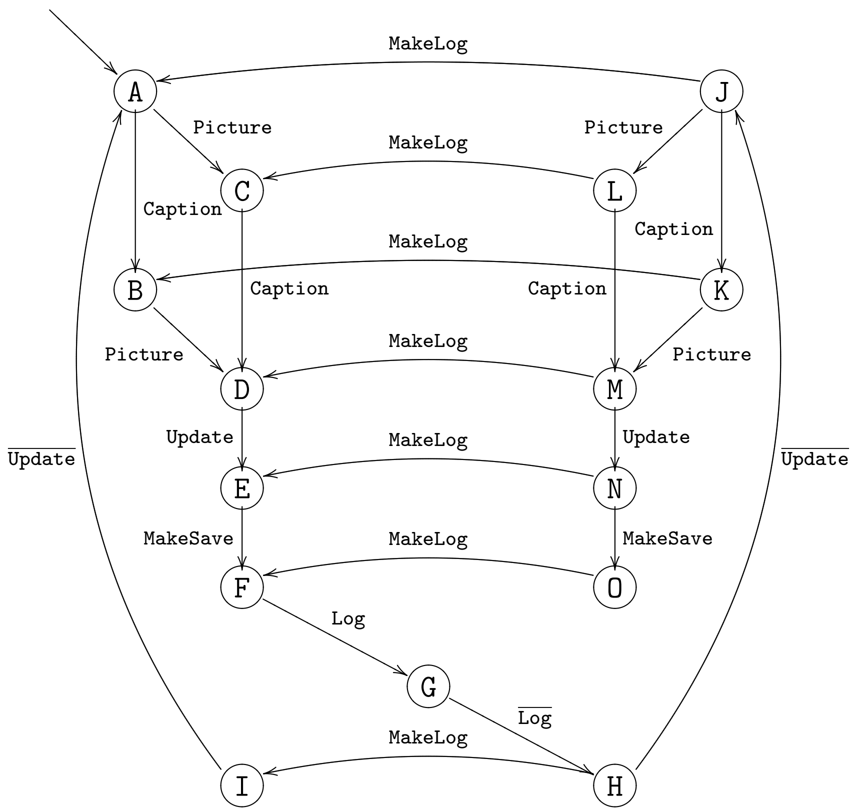

Figure 2: Model of the *Photo gallery* system

Example 3.2. *Consider a component-based system implementing a simple photo gallery, modelled as an LKS in Figure 2. The system consists of three components:* Controller, Updater *and* Logger, *with behavioural descriptions in Figure 1. The interface of the system is formed by* Controller, *which inserts photos with captions into the gallery. When provided with a picture and its caption (in any order),* Controller *asks* Updater *to update information in the gallery via action* `Update`, *which synchronizes with a corresponding action in* Updater. *In the model of the system in Figure 2,* Updater *gains focus and starts the update. It saves the changes first and then asks* Logger *to log the information.* Logger *responds to the* `Log` *call via* $\overline{\texttt{Log}}$ *and then completes the operation. Concurrently with the* `MakeLog` *operation of* Logger, Updater *returns response to* Controller, *which afterall takes all the components (and hence the system) to the initial setting.*

To make the LKS model of this system in Figure 2 complete, we need to add the set Ap *of atomic propositions and the state labelling function* $\mathcal{L}$. *In this case we choose the atomic propositions reflecting action enabledness. That is,* $\mathrm{Ap} = \{\mathcal{E}(a) \mid a \in \mathrm{Act}\}$ *where each* $\mathcal{E}(a)$ *represents a state proposition with the meaning "action* a *is enabled in this state". The labelling function is then given as* $\mathcal{L}(s) = \{\mathcal{E}(a) \mid \exists s' : s \xrightarrow{a} s'\}$ *associating with each state the actions enabled in that state. For example* $\mathcal{L}(\mathtt{A}) = \{\mathcal{E}(\mathtt{Picture}), \mathcal{E}(\mathtt{Caption})\}$.

We continue with defining the linear temporal logic that encompasses both state propositions and actions. This definition is equivalent to the definition in [6], it only slightly differs in notation.

Definition 3.3 (SE-LTL)**.** *Let* Act *be a set of actions,* Ap *a set of atomic propositions. The syntax of the* state/event LTL (SE-LTL for short) *formulae is defined inductively as:*

$$\varphi ::= \mathcal{P}(a) \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \mathbf{X}\,\varphi$$

where a *ranges over* Act *and* p *ranges over* Ap.
Let π *be a run of an LKS, the semantics of SE-LTL for runs is defined as:*

$$\begin{aligned}
\pi \models \mathcal{P}(a) \quad &\iff \quad \ell(\pi, 0) = a \\
\pi \models p \quad &\iff \quad p \in \mathcal{L}(\pi(0)) \\
\pi \models \neg\varphi \quad &\iff \quad \pi \not\models \varphi \\
\pi \models \varphi \wedge \psi \quad &\iff \quad \pi \models \varphi \textit{ and } \pi \models \psi \\
\pi \models \varphi \,\mathbf{U}\, \psi \quad &\iff \quad \exists k \geq 0 : \pi^k \models \psi \textit{ and } \forall j < k : \pi^j \models \varphi \\
\pi \models \mathbf{X}\,\varphi \quad &\iff \quad \pi^1 \models \varphi
\end{aligned}$$

Further, we say that an LKS M *satisfies* φ*, written as* $M \models \varphi$ *if for all initial runs* π *of* M*,* $\pi \models \varphi$.

Example 3.4. *Consider the LKS from Example 3.2 in Figure 2. Let* $\mathbf{F}\,\varphi$ *stand for* $\mathbf{true}\ \mathbf{U}\ \varphi$ *and* $\mathbf{G}\,\varphi$ *stand for* $\neg\,\mathbf{F}\,\neg\varphi$*. The property reflecting that an arbitrary number of pictures can be inserted into the album, can be stated in SE-LTL as* $\mathbf{G}\,\mathbf{F}\,\mathcal{P}(\texttt{Picture})$*. An example of a property using the state atomic propositions could be "whenever the action* `Picture` *becomes enabled, it is eventually executed", which is expressible as* $\mathbf{G}\,(\mathcal{E}(\texttt{Picture}) \Rightarrow \mathbf{F}\,\mathcal{P}(\texttt{Picture}))$*.*

It has been demonstrated in [6] that the automata-based approach for state-based LTL verification (see e.g. [10]) can be straightforwardly transformed into an automata-based verification method for SE-LTL with no extra cost.

4 Motivation

As mentioned in the introduction, to cope with the enormous size of real system models consisting of a large number of components it is necessary to employ reduction methods. The aim of such methods is to generate a reduced state space instead of the complete one while ensuring preservation of all required properties. One of such methods, the partial order reduction method, exploits the redundancies in the system caused by concurrent interleaving. When dealing with state-based LTL properties, the partial order reduction technique is built on the concept of stuttering equivalence [16]. Two runs of a system are considered to be stuttering equivalent if the only difference between them lies in sequential repetitions of states with identical labelling. The partial order reduction method then ensures that for each run of the system there is a stuttering equivalent run in the reduced state space. The subset of LTL properties that are preserved by this equivalence can be characterized syntactically: they are exactly those properties that can be written without the **X** operator.

To apply the partial order reduction method to SE-LTL, we need first to find a suitable concept that would play the role of stuttering equivalence for the state/event case. The above mentioned stuttering equivalence cannot be employed, as it considers state labelling only.

The first idea is to transfer the stuttering concept to actions. In the stuttering equivalence, consecutive states labelled with the same atomic propositions are ignored. Let us therefore consider an equivalence that ignores consecutive transitions labelled with the

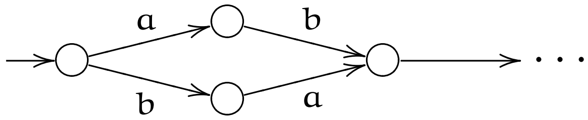

Figure 3: A typical situation for partial order reduction

same action and let us call this equivalence *action stuttering*. It implies that, for instance, two runs of the form $s_0, a, s_1, a, s_2, b, s_3, c, s_4, c, \ldots$ and $q_0, a, q_1, b, q_2, b, q_3, b, q_4, c, \ldots$ are action-stuttering equivalent. The advantage of this straightforward approach is that all formulae of action-based LTL not using the $\mathbf{X}$ operator are preserved by this equivalence. However, there is a number of arguments against this choice.

It is obvious that the partial order reduction method does not preserve this action-stuttering equivalence. Figure 3 shows a typical situation. If the transitions labelled with a and b are independent and one of them is invisible, the partial order reduction method traverses just one of the two runs $ab\ldots$ and $ba\ldots$ However, those two runs are not action-stuttering equivalent.

The problem is more fundamental. Consider a component-based system consisting of two components whose interaction is to be verified. Suppose we extend the system with an additional component that does not influence the communication of the original ones. A suitable substitution for stuttering equivalence should consider every run corresponding to the interaction behaviour of the original components equivalent regardless of interleaving with the third component. It is clear that the proposed action-stuttering equivalence does not satisfy this reasonable property.

We define a new equivalence, which, while still retaining the stuttering concept with respect to the state propositions, employs a different approach towards the transition labels (actions). This new equivalence enjoys the property that it is preserved by the partial order reduction method, thus allowing all the advantages of it. This comes at a cost. Contrary to state-based LTL, we do not have any syntactic characterization of SE-LTL formulae that are preserved by the new equivalence. However, we show that they can be elegantly described in terms of an adjusted weak version of SE-LTL.

5 State/Event Stuttering Equivalence

The main idea of the equivalence is that some of the actions are regarded as *interesting*. Transitions with noninteresting actions are then overlooked by the equivalence.

As we want to consider both actions and states, this idea is combined with the stuttering principle for state propositions, i.e. transitions which change state propositions we are interested in cannot be overlooked. In order to define the equivalence formally, we introduce the notions of a projection and a signature.

Definition 5.1 (projection, signature). *Let* $\pi = s_0, a_0, s_1, a_1, \ldots$ *be a run of LKS* $(S, \mathtt{Act}, \Delta, s_{init}, \mathtt{Ap}, \mathcal{L})$*, let* $\mathtt{Act}' \subseteq \mathtt{Act}$ *and* $\mathtt{Ap}' \subseteq \mathtt{Ap}$*. Let* τ *be a new symbol,* $\tau \notin \mathtt{Act}$*. A* projection of π onto $\mathtt{Act}'$and $\mathtt{Ap}'$ *is defined as*

$$\mathsf{pr}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi) = E_0, b_0, E_1, b_1, \ldots$$

where $E_i = \mathcal{L}(s_i) \cap \mathtt{Ap}'$*,* b_i *is equal to* τ *whenever* $a_i \notin \mathtt{Act}'$*and* $b_i = a_i$ *otherwise.*

Furthermore, a signature of π with respect to $\mathtt{Act}'$ and $\mathtt{Ap}'$*, denoted as* $\mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi)$ *is defined as the (finite or infinite) alternating sequence of sets of atomic propositions and actions that arises from the projection of* π *onto* $\mathtt{Act}'$*and* $\mathtt{Ap}'$ *by replacing every maximal subsequence of the form* $E_i, \tau, E_{i+1}, \tau, \ldots$*, where* $E_i = E_{i+1} = \cdots$*, with just* E_i*.*

We also define a tail *of* π *with respect to* $\mathtt{Act}'$*and* $\mathtt{Ap}'$*, denoted as* $\mathsf{tail}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi)$*, as the maximal subrun* π^k *of* π *such that* $\mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi) = E_0, \alpha, \mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi^k)$ *for some* $E_0 \subseteq \mathtt{Ap}'$*and* $\alpha \in \mathtt{Act}' \cup \{\tau\}$*. Note that the tail is undefined for runs whose signature is of the form* E_0*.*

Example 5.2. *Let* $\pi = \mathtt{A}, \mathtt{Caption}, \mathtt{B}, \mathtt{Picture}, \mathtt{D}, \mathtt{Update}, \mathtt{E}, \mathtt{MakeSave}, \mathtt{F}, \ldots$ *and let* $\mathtt{Act}' = \{\mathtt{Update}\}$ *and* $\mathtt{Ap}' = \{\mathcal{E}(\mathtt{Picture})\}$*. Then the projection of* π *onto* $\mathtt{Act}'$ *and* $\mathtt{Ap}'$ *is* $\mathsf{pr}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi) = \{\mathcal{E}(\mathtt{Picture})\}, \tau, \{\mathcal{E}(\mathtt{Picture})\}, \tau, \emptyset, \mathtt{Update}, \emptyset, \tau, \emptyset, \ldots$ *and its signature is* $\mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi) = \{\mathcal{E}(\mathtt{Picture})\}, \tau, \emptyset, \mathtt{Update}, \emptyset, \ldots$

Definition 5.3 (state/event stuttering equivalence). *Let* π *and* σ *be two runs, let* $\mathtt{Act}'$ *be a set of actions,* $\mathtt{Ap}'$ *a set of atomic propositions. We say that* π *and* σ *are* state/event stuttering equivalent with respect to $\mathtt{Act}'$and $\mathtt{Ap}'$*, denoted as* $\pi \equiv^{\mathtt{Ap}'}_{\mathtt{Act}'} \sigma$*, if they have the same signatures, i.e.* $\mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\pi) = \mathsf{sig}^{\mathtt{Ap}'}_{\mathtt{Act}'}(\sigma)$*.*

Two LKSs are said to be state/event stuttering equivalent with respect to $\mathtt{Act}'$*and* $\mathtt{Ap}'$*, if for each run of one LKS there is a state/event stuttering equivalent run of the other and vice versa.*

Stuttering equivalence is a special case of state/event stuttering equivalence for $\mathtt{Act}' = \emptyset$.

Definition 5.4 (weak SE-LTL). *Let* $\mathtt{Act}$ *be a set of actions,* $\mathtt{Ap}$ *a set of atomic propositions and let* $\mathtt{Act}' \subseteq \mathtt{Act}$*. We define the* weak state/event LTL with respect to $\mathtt{Act}'$*, wSE-LTL for short, as follows. The syntax of the formulae is defined inductively as:*

$$\varphi ::= \widetilde{\mathcal{P}}(a) \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathbf{U}\, \varphi_2 \mid \widetilde{\mathbf{X}}\, \varphi \mid \varphi_1 \,\mathbf{U}_a\, \varphi_2$$

where a *ranges over* Act' *and* p *ranges over* Ap.

Let π *be a run of an LKS. The semantics for runs is defined inductively as*

$$\begin{array}{lcl}
\pi \models \widetilde{\mathcal{P}}(a) & \iff & \exists k \geq 0 : \ell(\pi,k) = a \textit{ and } \forall j < k : \ell(\pi,j) \notin \mathrm{Act}' \\
\pi \models p & \iff & p \in \mathcal{L}(\pi(0)) \\
\pi \models \neg\varphi & \iff & \pi \not\models \varphi \\
\pi \models \varphi \wedge \psi & \iff & \pi \models \varphi \textit{ and } \pi \models \psi \\
\pi \models \varphi \,\mathbf{U}\, \psi & \iff & \exists k \geq 0 : \pi^k \models \psi \textit{ and } \forall j < k : \pi^j \models \varphi \\
\pi \models \widetilde{\mathbf{X}}\, \varphi & \iff & \exists k \geq 0 : \ell(\pi,k) \in \mathrm{Act}', \forall j < k : \ell(\pi,j) \notin \mathrm{Act}' \textit{ and } \pi^{k+1} \models \varphi \\
\pi \models \varphi \,\mathbf{U}_a\, \psi & \iff & \exists k \geq 0 : \ell(\pi,k) = a, \pi^{k+1} \models \psi \textit{ and } \forall j < k+1 : \pi^j \models \varphi
\end{array}$$

The main difference between SE-LTL and wSE-LTL is in the semantics of the next and action operators. While $\mathcal{P}(a)$ states that "the first action is a", $\widetilde{\mathcal{P}}(a)$ states that "the first *interesting* action is a". The formula $\mathbf{X}\,\varphi$ states that "in the next step, φ holds", while the formula $\widetilde{\mathbf{X}}\,\varphi$ states that "after the next *interesting action*, φ holds". Here, *interesting* means "from Act'". Note that the semantics of wSE-LTL depends on the choice of Act' and different runs may satisfy a formula of wSE-LTL depending on this choice. When specifying properties in wSE-LTL, it is therefore assumed that a pair (φ, Act') is given instead of just the formula.

Actually, the definition of $\widetilde{\mathbf{X}}\,\varphi$ is of the form "there is a next interesting action *and* after this action, φ holds". We could also sometimes be interested in stating the property that "*if* there is a next interesting action, *then* after this action, φ holds". Nevertheless, this alternative $\widehat{\mathbf{X}}\,\varphi$ operator can be defined as a derived operator: $\widehat{\mathbf{X}}\,\varphi := \neg\,\widetilde{\mathbf{X}}\neg\varphi$.

Additionally, wSE-LTL has a new operator $\mathbf{U}_a$. The motivation is to express properties like "atomic proposition p holds until action a happens". In SE-LTL, this can be expressed with $p\ \mathbf{U}\ (p \wedge \mathcal{P}(a))$. In wSE-LTL this is no longer possible as the semantics of $\widetilde{\mathcal{P}}(a)$ is different and the intuitive solution $p\ \mathbf{U}\ (p \wedge \widetilde{\mathcal{P}}(a))$ holds even for runs that do not satisfy the original property, e.g. a run with signature $\{p\}, \tau, \{\neg p\}, a, \ldots$ Thanks to the $\mathbf{U}_a$ operator, this property is expressible as $p\ \mathbf{U}_a\ \mathbf{true}$. The $\mathbf{U}_a$ operator is not needed in the action-based fragment of wSE-LTL. The reader may verify that every formula $\varphi\,\mathbf{U}_a\,\psi$, where φ and ψ both do not use state atomic propositions, is equivalent to $\varphi\ \mathbf{U}\ (\widetilde{\mathcal{P}}(a) \wedge \varphi \wedge \widetilde{\mathbf{X}}\,\psi)$.

We are now ready to present the main result of this section, namely that the properties expressible in weak SE-LTL are preserved by the state/event stuttering equivalence.

Lemma 5.5 (equivalence of subruns). *Let π and σ be two runs such that $\pi \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma$. Then there is a nondecreasing function $f\colon \mathbb{N}_0 \to \mathbb{N}_0$ such that $f(0) = 0$ and for each $i \geq 0$, $\pi^i \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(i)}$. Moreover, for each i, the following holds:*

$$\sigma^{f(i)} \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(i)+1} \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \cdots \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(i+1)-1}$$

Proof. We define the function f inductively as follows:

- $f(0) = 0$. Obviously $\pi^0 = \pi \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma = \sigma^0$.
- Let f be defined for $0, \ldots, n$ and we want to define $f(n+1)$. There are two cases:
  - $\ell(\pi, n) \notin \mathrm{Act}'$ and $\mathcal{L}(\pi(n)) \cap \mathrm{Ap}' = \mathcal{L}(\pi(n+1)) \cap \mathrm{Ap}'$. Then $f(n+1) = f(n)$. It is clear that $\pi^{n+1} \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \pi^n \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(n)} = \sigma^{f(n+1)}$.
  - $\ell(\pi, n) \in \mathrm{Act}'$ or $\mathcal{L}(\pi(n)) \cap \mathrm{Ap}' \neq \mathcal{L}(\pi(n+1)) \cap \mathrm{Ap}'$. Then $\pi^{n+1} = \mathsf{tail}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\pi^n)$. We define $f(n+1)$ such that $\sigma^{f(n+1)} = \mathsf{tail}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\sigma^{f(n)})$. It is easily seen that if two runs are state/event stuttering equivalent, their tails also are, therefore $\pi^{n+1} = \mathsf{tail}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\pi^n) \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \mathsf{tail}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\sigma^{f(n)}) = \sigma^{f(n+1)}$.

It should be clear that the function f is nondecreasing. It remains to show that

$$\sigma^{f(i)} \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(i)+1} \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \cdots \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma^{f(i+1)-1}$$

for all i. If this condition were violated then both the signature of $\sigma^{f(i)}$ and the signature of its tail would differ from the signature of $\sigma^{f(i+1)}$. Clearly, from the construction of f this is not possible. $\square$

Theorem 5.6 (equivalence preserves properties). *Let Act' be a set of actions, Ap' a set of atomic propositions, and let π and σ be two runs such that $\pi \equiv_{\mathrm{Act}'}^{\mathrm{Ap}'} \sigma$. Then for each formula φ of wSE-LTL with respect to Act' such that it only contains atomic propositions from Ap', $\pi \models \varphi$ if and only if $\sigma \models \varphi$.*

Proof. Let $\mathsf{sig}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\pi) = \mathsf{sig}_{\mathrm{Act}'}^{\mathrm{Ap}'}(\sigma)$. The proof follows by induction on the formula. We only show three interesting cases and only one direction, i.e. that $\pi \models \varphi$ implies $\sigma \models \varphi$. The other cases are evident and the other direction follows from the symmetry of the equivalence.

- $\pi \models \psi_1 \mathbf{U} \psi_2$. Then there is some k such that $\pi^k \models \psi_2$ and for each $j < k$, $\pi^j \models \psi_1$. But then Lemma 5.5 together with the induction hypothesis implies that $\sigma^{f(k)} \models \psi_2$ and for each $i < f(k)$, $\sigma^i \models \psi_1$, that is $\sigma \models \psi_1 \mathbf{U} \psi_2$.

- $\pi \models \mathbf{X}\,\psi$. Then there is some k such that $\ell(\pi, k)$ is the first interesting action on π and $\pi^{k+1} \models \psi$. From the state/event stuttering equivalence, there has to be some j such that $\ell(\sigma, j)$ is the first interesting action on σ and $\mathsf{sig}^{Ap'}_{Act'}(\sigma^{j+1}) = \mathsf{sig}^{Ap'}_{Act'}(\pi^{k+1})$ is the part of the original signature that start after the first interesting action. Thus $\sigma \models \mathbf{X}\,\psi$.

- $\pi \models \psi_1\, \mathbf{U}_a\, \psi_2$. Then there is some k such that $\ell(\pi, k) = a$, $\pi^{k+1} \models \psi_2$ and for all $j < k+1$, $\pi^j \models \psi_1$. As in the case of $\mathbf{U}$, it follows from Lemma 5.5 that $\sigma^{f(k+1)} \models \psi_2$ and $\sigma^i \models \psi_1$ for all $i < f(k+1)$. We need to show that $\ell(\sigma, f(k+1)-1) = a$. However, we know that $\mathsf{sig}^{Ap'}_{Act'}(\pi^k) = E, a, \mathsf{sig}^{Ap'}_{Act'}(\pi^{k+1})$ for some E and we know from Lemma 5.5 that $\sigma^{f(k+1)-1} \equiv^{Ap'}_{Act'} \sigma^{f(k)} \equiv^{Ap'}_{Act'} \pi^k$. Therefore, $\mathsf{sig}^{Ap'}_{Act'}(\sigma^{f(k+1)-1}) = E, a, \mathsf{sig}^{Ap'}_{Act'}(\sigma^{f(k+1)})$ and thus $\ell(\sigma, f(k+1)-1) = a$ and $\sigma \models \psi_1\, \mathbf{U}_a\, \psi_2$. $\square$

What remains is to show that wSE-LTL is indeed a weak version of SE-LTL, i.e. that all properties expressible in wSE-LTL are also expressible in SE-LTL. The following theorem states that every formula of wSE-LTL can be translated in linear time to an equivalent formula of SE-LTL. This way the verification problem for wSE-LTL can be reduced to the verification problem for SE-LTL, which is solvable in a way similar to the standard LTL verification as described in [6].

Theorem 5.7 (embedding of wSE-LTL into SE-LTL). *Every formula φ of weak SE-LTL with respect to Act' can be translated to a formula $T(\varphi)$ of SE-LTL such that for each π, $\pi \models \varphi$ if and only if $\pi \models T(\varphi)$.*

Proof. We define an auxiliary formula $\xi := \bigwedge_{a \in Act'} \neg\mathcal{P}(a)$. The translation is defined inductively as follows:

$$
\begin{aligned}
T(p) &:= p \\
T(\widetilde{\mathcal{P}}(a)) &:= \xi\, \mathbf{U}\, \mathcal{P}(a) \\
T(\widetilde{\mathbf{X}}\, \varphi) &:= \xi\, \mathbf{U}\, (\neg\xi \wedge \mathbf{X}\, T(\varphi)) \\
T(\varphi\, \mathbf{U}\, \psi) &:= T(\varphi)\, \mathbf{U}\, T(\psi) \\
T(\varphi\, \mathbf{U}_a\, \psi) &:= T(\varphi)\, \mathbf{U}\, (\mathcal{P}(a) \wedge T(\varphi) \wedge \mathbf{X}\, T(\psi)) \\
T(\varphi \wedge \psi) &:= T(\varphi) \wedge T(\psi) \\
T(\neg\varphi) &:= \neg T(\varphi)
\end{aligned}
$$

The correctness of the construction is proved by induction. The interesting cases are $\widetilde{\mathcal{P}}$, $\widetilde{\mathbf{X}}$ and $\mathbf{U}_a$.

$$
\begin{aligned}
\pi \models \widetilde{\mathcal{P}}(a) &\iff \exists k \geq 0 : \ell(\pi, k) = a \text{ and } \forall j < k : \ell(\pi, j) \notin \mathrm{Act}' \\
&\iff \exists k \geq 0 : \pi^k \models \mathcal{P}(a) \text{ and } \forall j < k : \pi^j \models \xi \\
&\iff \pi \models \xi \,\mathbf{U}\, \mathcal{P}(a) \\
\pi \models \widetilde{\mathbf{X}}\, \varphi &\iff \exists k \geq 0 : \ell(\pi, k) \in \mathrm{Act}', \forall j < k : \ell(\pi, j) \notin \mathrm{Act}' \text{and } \pi^{k+1} \models \varphi \\
&\iff \exists k \geq 0 : \pi^k \models \neg\xi, \forall j < k : \pi^j \models \xi \text{ and } \pi^{k+1} \models T(\varphi) \\
&\iff \exists k \geq 0 : \pi^k \models \neg\xi \wedge \mathbf{X}\, T(\varphi) \text{ and } \forall j < k : \pi^j \models \xi \\
&\iff \pi \models \xi \,\mathbf{U}\, (\neg\xi \wedge \mathbf{X}\, T(\varphi)) \\
\pi \models \varphi \,\mathbf{U}_a\, \psi &\iff \exists k \geq 0 : \ell(\pi, k) = a, \pi^{k+1} \models \psi \text{ and } \forall j < k+1 : \pi^j \models \varphi \\
&\iff \exists k \geq 0 : \pi^k \models \mathcal{P}(a), \pi^{k+1} \models T(\psi) \\
&\qquad \text{and } \forall j < k+1 : \pi^j \models T(\varphi) \\
&\iff \exists k \geq 0 : \pi^k \models \mathcal{P}(a), \pi^k \models \mathbf{X}\, T(\psi), \pi^k \models T(\varphi) \\
&\qquad \text{and } \forall j < k : \pi^j \models T(\varphi) \\
&\iff \pi \models T(\varphi) \,\mathbf{U}\, (\mathcal{P}(a) \wedge \mathbf{X}\, T(\psi) \wedge T(\varphi))
\end{aligned}
$$

The remaining cases are straightforward. □

5.1 Characterization of invariant SE-LTL properties

We have shown that wSE-LTL is preserved by state/event stuttering equivalence and can be embedded into SE-LTL. Thus, wSE-LTL can be seen as a characterization of some SE-LTL properties that are preserved by state/event stuttering equivalence (we use the term *state/event stutter-invariant* for such properties in the following). We now show that this characterization is exact, i.e. that all state/event stutter-invariant SE-LTL properties are expressible in wSE-LTL. The proof follows the method of [16].

Definition 5.8. *A run π is* state/event stutter-free, *if for each $i \geq 0$ one of the following holds:*

- $\ell(\pi, i) \in \mathrm{Act}'$ *(ith transition is labelled by interesting action)*
- $\mathcal{L}(\pi(i)) \cap \mathrm{Ap}' \neq \mathcal{L}(\pi(i+1)) \cap \mathrm{Ap}'$ *(ith transition changes the state labelling)*
- $\ell(\pi, j) \notin \mathrm{Act}'$ *and* $\mathcal{L}(\pi(j)) \cap \mathrm{Ap}' = \mathcal{L}(\pi(j+1)) \cap \mathrm{Ap}'$ *for all* $j \geq i$ *(nothing interesting ever happens from ith position onwards)*

It is clear that a state/event stutter-free run is a unique representant of its state/event stuttering equivalence class. Note that an arbitrary subrun of a state/event stutter-free run is also state/event stutter-free.

In the following, we assume that $Ap' = \{p_1, \dots, p_n\}$ and let N be the set of all subsets of Ap', i.e. $N = 2^{Ap'}$. For each $\nu \in N$, let β_ν be the formula $\alpha_1 \wedge \cdots \wedge \alpha_n$ where $\alpha_j = p_j$ if $p_j \in \nu$ and $\alpha_j = \neg p_j$ otherwise. If $Ap' = \emptyset$ then $N = \{\emptyset\}$ and $\beta_\nu = \mathbf{true}$. Thus, β_ν holds in precisely those states whose valuation is equal to ν.

Lemma 5.9. *Let π be a state/event stutter-free run. Then $\ell(\pi, 0) \in Act'$ if and only if $\pi \models \bigvee_{\nu \in N} \big(\bigvee_{a \in Act'} (\beta_\nu \, \mathbf{U}_a \, \mathbf{true})\big)$.*

Proof. If $\pi = s_0, a, \dots$ where $a \in Act'$ then clearly $\pi \models \beta_\nu \, \mathbf{U}_a \, \mathbf{true}$ where ν is the set of all state propositions true at s_0.

The other direction is proved by contradiction. Suppose that $\ell(\pi, 0) \notin Act'$ and that π satisfies the formula in the lemma. Then, there is some ν and a such that $\pi \models \beta_\nu \mathbf{U}_a \mathbf{true}$. Therefore, $\mathsf{pr}^{Ap'}_{Act'}(\pi)$ has to be of the form $E_0, \tau, E_1, \dots, E_i, a, E_{i+1}, \dots$ where both E_0 and E_1 are equal to ν. But then, π is not state/event stutter-free, which is a contradiction. $\square$

Theorem 5.10. *Every state/event stutter-invariant property expressible in SE-LTL is expressible in wSE-LTL.*

Proof. We will show that for every SE-LTL formula φ there exists a wSE-LTL formula $\tau(\varphi)$ that agrees with φ on all state/event stutter-free runs. Clearly this implies the theorem.

The formula $\tau(\varphi)$ is defined inductively as follows. The straightforward parts are $\tau(p) = p$ for $p \in Ap'$, $\tau(\neg\varphi) = \neg\tau(\varphi)$, $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$ and $\tau(\varphi \mathbf{U} \psi) = \tau(\varphi) \mathbf{U} \tau(\psi)$. These choices are obviously correct. The more difficult parts are that of $\mathcal{P}(a)$ and $\mathbf{X}\,\varphi$. They are dealt with as follows:

$$\tau(\mathcal{P}(a)) := \widetilde{\mathcal{P}}(a) \wedge \bigvee_{\nu \in N} \beta_\nu \, \mathbf{U}_a \, \mathbf{true}$$
$$\tau(\mathbf{X}\,\varphi) := \psi_1 \vee \psi_2 \vee \psi_3$$

where

$$\psi_1 := \bigvee_{\nu \in N} \left(\mathbf{G}\, \beta_\nu \wedge \neg \widetilde{\mathbf{X}}\, \mathbf{true} \wedge \tau(\varphi) \right)$$

$$\psi_2 := \bigvee_{\nu \in N} \bigvee_{a \in Act'} \left(\beta_\nu \, \mathbf{U}_a \, \mathbf{true} \wedge \widetilde{\mathbf{X}}\, \tau(\varphi) \right)$$

$$\psi_3 := \neg \bigvee_{\nu \in N} \bigvee_{a \in Act'} (\beta_\nu \, \mathbf{U}_a \, \mathbf{true}) \wedge \bigvee_{\nu \in N} \bigvee_{\nu' \in N \setminus \{\nu\}} (\beta_\nu \wedge (\beta_\nu \, \mathbf{U} \, (\beta_{\nu'} \wedge \tau(\varphi))))$$

What remains is to prove the correctness of this choice. The choice of $\tau(\mathcal{P}(a))$ is evidently correct by an argument similar to that of Lemma 5.9.

Let us now suppose that $\pi \models \mathbf{X}\, \varphi$. We want to show that then $\pi \models \tau(\mathbf{X}\, \varphi)$. It follows from the induction hypothesis that $\pi^1 \models \tau(\varphi)$. As π is state/event stutter-free run, it has to be of one of the following three forms:

(i) $\mathsf{pr}^{Ap'}_{Act'}(\pi) = E_0, \tau, E_1, \tau, E_2, \tau, \ldots$, where $E_0 = E_1 = E_2 = \ldots$, i.e. nothing interesting ever happens on π. In this case it is obvious that $\pi \models \mathbf{G}\, \beta_\nu$ where $\nu = E_0$ and $\pi \models \neg \widetilde{\mathbf{X}}\, \mathbf{true}$. As $\mathsf{pr}^{Ap'}_{Act'}(\pi) = \mathsf{pr}^{Ap'}_{Act'}(\pi^1)$, we also know that $\pi \models \tau(\varphi)$. Thus $\pi \models \psi_1$ and therefore $\pi \models \tau(\mathbf{X}\, \varphi)$.

(ii) $\mathsf{pr}^{Ap'}_{Act'}(\pi) = E_0, a, E_1, \ldots$, where $a \in Act'$, i.e. first transition of π has interesting label. Then, it follows from Lemma 5.9 that $\pi \models \beta_\nu \mathbf{U}_a \mathbf{true}$ for some ν and a. From the definition of $\widetilde{\mathbf{X}}$ and the fact that $\pi^1 \models \tau(\varphi)$ it then follows that $\pi \models \widetilde{\mathbf{X}}\, \tau(\varphi)$. Thus $\pi \models \psi_2$ and therefore $\pi \models \tau(\mathbf{X}\, \varphi)$.

(iii) $\mathsf{pr}^{Ap'}_{Act'}(\pi) = E_0, \tau, E_1, \ldots$, where $E_0 \neq E_1$, i.e. first transition of π has noninteresting label but changes state labelling. In this case $\pi \models \beta_\nu \wedge (\beta_\nu \, \mathbf{U} \, (\beta_{\nu'} \wedge \tau(\varphi)))$, where $\nu = E_0$ and $\nu' = E_1$. That $\pi \models \neg(\beta_\nu \, \mathbf{U}_a \, \mathbf{true})$ for every value of ν and a follows from Lemma 5.9. Thus $\pi \models \psi_3$ and therefore $\pi \models \tau(\mathbf{X}\, \varphi)$.

For the other direction of the proof, suppose that $\pi \models \tau(\mathbf{X}\, \varphi)$. We want to show that $\pi \models \mathbf{X}\, \varphi$. The induction hypothesis is that whenever a state/event stutter-free run satisfies $\tau(\varphi)$, it also satisfies φ. There are three cases:

(a) $\pi \models \psi_1$. As $\pi \models \mathbf{G}\, \beta_\nu$ and $\pi \models \neg \widetilde{\mathbf{X}}\, \mathbf{true}$, the run π has to be of the form (i). As $\pi \models \tau(\varphi)$ and $\pi = \pi^1$ then clearly $\pi^1 \models \varphi$ and thus $\pi \models \mathbf{X}\, \varphi$.

(b) $\pi \models \psi_2$. By Lemma 5.9, π has to be of the form (ii), i.e. its first transition has to be labelled by an interesting action. But then $\pi \models \widetilde{\mathbf{X}}\, \tau(\varphi)$ implies $\pi^1 \models \tau(\varphi)$ and thus $\pi \models \mathbf{X}\, \varphi$.

(c) $\pi \models \psi_3$. By Lemma 5.9, π has to be of either the form (i) or (iii). As $\pi \models \beta_\nu \wedge (\beta_\nu \mathbf{U} (\beta_{\nu'} \wedge \tau(\varphi)))$ for some ν and ν' such that $\nu \neq \nu'$, it cannot be of the form (i). Thus, it has to be that $\mathsf{pr}^{Ap'}_{Act'}(\pi) = E_0, \tau, E_1, \ldots$ where $E_0 \neq E_1$. Then clearly $\nu = E_0$ and $\pi^1 \models \beta_{\nu'} \wedge \tau(\varphi)$ as it cannot be the case that $\pi^1 \models \beta_\nu$ because π is state/event stutter-free. But then $\pi^1 \models \varphi$ and $\pi \models \mathbf{X}\,\varphi$. This completes the proof. □

Although the construction in the above proof is exponential in worst case, the main significance of this result is that using weak SE-LTL comes without loss of expressiveness over SE-LTL if we want to use state/event stutter-invariant properties. Moreover, the construction justifies the choice of wSE-LTL operators, namely the $\widetilde{\mathcal{P}}$, $\widetilde{\mathbf{X}}$ and $\mathbf{U}_a$ operators, which made the construction possible. Note that if any of these were excluded from the logic, it would be less expressive and the above result would not hold.

6 Partial Order Reduction

The goal of this section is to show that the partial order reduction method can be applied to LKSs so that the reduced LKS remains state/event stuttering equivalent. At first, we summarize the basics of the partial order reduction method. While presenting the method we follow the explication from [7]. Consequently, we explain how the method can be applied to SE-LTL.

Definition 6.1 (state transition system)**.** *A* state transition system *is a triple* (S, T, s_{init}) *where* S *is a set of states,* s_{init} *is an initial state and* T *is a set of transitions such that for each* $\alpha \in T$, $\alpha \subseteq S \times S$. *Further, for each* $\alpha \in T$ *and for each state* $s \in S$ *there is at most one* $s' \in S$ *such that* $(s, s') \in \alpha$.

An initial transition path *of a state transition system is an infinite sequence* $\alpha_0\alpha_1\ldots$ *such that there are states* $s_0, s_1, \ldots$ *satisfying* $s_0 = s_{init}$ *and for all* i, $(s_i, s_{i+1}) \in \alpha_i$.

The idea of the ample set method [14, 15] is to construct a reduced state space by choosing a smaller set of successors at each state. Instead of exploring all successors from a given state, denoted as *enabled*(s), we explore only successors from *ample*$(s) \subseteq$ *enabled*(s).

Theorem 6.2. *[7] Let* M *be a state transition system and let* $V \subseteq T$ *be an arbitrary set of* visible *transitions. Let* M' *be the reduced system constructed using the ample set partial order algorithm. Then for each initial transition path* π *from* M *there is an initial transition path* σ *in* M' *such that* π *and* σ *have the same sequence of visible transitions.*

Proof. The theorem follows from the proof of Theorem 12 in [7]. $\square$

Now we present a transformation of an LKS to a state transition system. From now on, we fix the LKS $\mathcal{K} = (S, Act, \Delta, s_{init}, Ap, \mathcal{L})$, and two sets, $Act' \subseteq Act$ and $Ap' \subseteq Ap$ of interesting actions and interesting atomic propositions respectively. We need the notions of invisibility and proper transition partition first.

Definition 6.3 (invisibility). *A transition* $(q, a, r) \in \Delta$ *is called* invisible *if* $a \notin Act'$ *and* $\mathcal{L}(q) \cap Ap' = \mathcal{L}(r) \cap Ap'$. *A transition is called* visible *if it is not invisible.*

Definition 6.4 (proper transition partition). *An indexed set* $P = \{\Delta_i \mid i \in I\}$ *is called a* proper transition partition *if the following holds:*

- P *is a partition of* Δ, *i.e.* $\bigcup_{i \in I} \Delta_i = \Delta$ *and* $\Delta_i \cap \Delta_j = \emptyset$ *for all* $i \neq j$.
- P *preserves actions, i.e.* $\forall i \in I \,.\, \exists a \in Act : \Delta_i \subseteq S \times \{a\} \times S$.
- P *preserves invisibility, i.e. for all* i *either all transitions from* Δ_i *are invisible or none is.*
- P *is deterministic, i.e. for all* i *and for all* $s \in S$, *there is at most one* $s' \in S$ *such that* $(s, a, s') \in \Delta_i$.

We now present a transformation of LKS $\mathcal{K} = (S, Act, \Delta, s_{init}, Ap, \mathcal{L})$ with a proper transition partition P into a state transition system $\mathcal{M} = (S, T, s_{init})$. Let $T = \{\alpha_i \mid i \in I\}$ and $\alpha_i = \{(s, s') \mid (s, a, s') \in \Delta_i\}$. The set of visible transitions $V \subseteq T$ is defined as $V = \{\alpha_i \mid \text{transitions in } \Delta_i \text{ are visible}\}$. We denote $\Delta(\alpha_i) = \Delta_i$ the underlying partition set for α_i. Let $\alpha = \alpha_0\alpha_1 \ldots$ be an initial transition path of the state transition system (S, T, s_{init}). We assign to α an initial run $s_0, a_0, s_1, a_1, \ldots$ where $s_0 = s_{init}$, $(s_i, a_i, s_{i+1}) \in \Delta(\alpha_i)$ for $i \geq 0$. Clearly, the assigned initial run is a unique initial run of the LKS due to the nature of the proper transition partition.

Theorem 6.5. *Let* $\alpha = \alpha_0\alpha_1 \ldots$ *and* $\beta = \beta_0\beta_1 \ldots$ *be two initial transition paths of the state transition system* (S, T, s_{init}). *Let* π *and* σ *be the initial runs assigned to* α *and* β, *respectively. Then* α *and* β *have the same sequence of visible transitions if and only if* $\pi \equiv^{Ap'}_{Act'} \sigma$.

Proof. Invisible transitions of the state transition system represent transitions of the original LKS such that they do not contribute to the runs' signatures and vice versa. $\square$

Corollary 6.6. *Let* M *be an LKS and let* $Act' \subseteq Act$ *and* $Ap' \subseteq Ap$. *Let* M' *be the reduced system constructed using the ample set partial order algorithm on the state transition system that is created as discussed above. Then for each initial run* π *of* M *there is an initial run* σ *of* M' *such that* $\pi \equiv^{Ap'}_{Act'} \sigma$.

Three things have to be supplied to the partial order reduction algorithm along with the LKS to be verified in order for the method to work. They are the proper transition partition, the set of interesting atomic propositions $\mathtt{Ap}'$ and the set of interesting actions $\mathtt{Act}'$.

A proper transition partition can be constructed automatically from some additional information of the structure of the LKS in question. Take for instance that the LKS represents a component-based system made from a number of smaller components such as the LKS in Example 3.2. A proper transition partition can be constructed in such a way that all transitions of the system that correspond with one transition of a smaller component constitute exactly one set of the partition.

A set of interesting atomic propositions is acquired from the verified formula. It is constructed as the set of all atomic propositions present in the formula. A set of interesting actions, however, has to be supplied by the user by hand. This set has to be a part of the property specification, as noted in Section 5, nonetheless. The cardinality of this set can affect the effectivity of the reduction method. It is thus desirable to specify as small set of interesting actions as possible, bearing in mind the intended semantics of the verified formula.

The partial order reduction itself can then be done on the fly during the automata-based verification process. Known methods and implementations can be used for this purpose, see e.g. [3].

Example 6.7. *Consider the LKS from Example 3.2. If we choose* $\mathtt{Ap}'$ *to be an arbitrary subset of* $\mathtt{Ap}$ *and* $\mathtt{Act}'$ *to be an arbitrary subset of* $\mathtt{Act}$ *such that* $\mathtt{MakeLog} \notin \mathtt{Act}'$ *then the state space of the reduced LKS will look as depicted in Figure 6. In this case* $\mathit{ample}(\mathtt{s}) = \mathit{enabled}(\mathtt{s})$ *for all states except* $\mathtt{H}$ *and* $\mathit{ample}(\mathtt{H}) = \{\mathtt{I}\}$.

7 Conclusion and Future Work

The paper introduces a partial order reduction technique for state/event LTL. The technique is based on a new stuttering equivalence, which is able to reflect both state and transition labels while regarding both with a different principle to closely fit their nature. On the level of states, the stuttering concentrates on changes in assigned atomic propositions along a run, whereas in the case of actions, the interesting events are observed at every single occurrence of an action representing interesting behaviour of the system, which is stated explicitly with respect to the verified property for instance. The

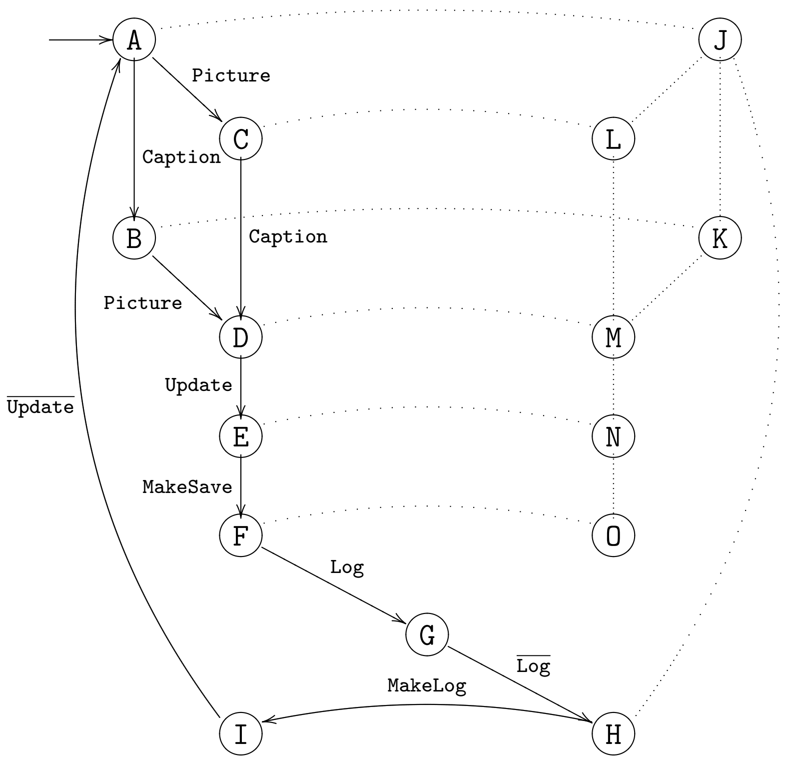

Figure 4: Example from Figure 2 after partial order reduction

paper moreover gives the characterization of the state/event LTL properties preserved by the equivalence, and summarizes the attributes into the definition of its fragment, called weak state/event LTL. This fragment is preserved by the equivalence while staying strong enough to reflect interesting component-specific properties, discussed at the beginning of this paper.

After introducing both the equivalence and the corresponding logic fragment, we discuss the partial order reduction technique. We show that the partial order reduction task for the state/event case can be translated into the state-based case via providing existing algorithms with a modified definition of transition invisibility reflecting the discussed specifics. The advantage of such translation is that known algorithms may be used for solving the problem. Moreover, as the efficiency of the partial order reduction method has been experimentally verified on a number of case studies, our approach can be expected to be efficient as well.

Future work. Our ongoing and future aims are connected to the employment of the technique into our framework for formal verification of component-based systems [8] based upon the formalism of Component-Interaction automata [18, 4]. The framework, in connection with the model checking tool DiVinE [1], recently helped us to perform an extensive verification case study [2], which uncovered the need of a partial order reduction method for state/event systems not found elsewhere. Currently we work on the implementation and plan to perform a detailed experimental case study using our framework and Component-Interaction automata as an underlying formalism. The technique, however, is general and independent on the application we aim it for.

References

[1] J. Barnat, L. Brim, I. Černá, P. Moravec, P. Ročkai, and P. Šimeček. DiVinE – a tool for distributed verification. In *Proceedings of CAV'06*, volume 4144 of *LNCS*, pages 278–281. Springer-Verlag, 2006.

[2] N. Beneš, I. Černá, J. Sochor, P. Vařeková, and B. Zimmerova. A case study in parallel verification of component-based systems. In *Proceedings of PDMC'08*, pages 35–51, 2008.

[3] D. Bosnacki, S. Leue, and A. Lluch-Lafuente. Partial-order reduction for general state exploring algorithms. In *Proceedings of SPIN'06*, volume 3925 of *LNCS*, pages 271–287. Springer-Verlag, 2006.

[4] L. Brim, I. Černá, P. Vařeková, and B. Zimmerova. Component-Interaction automata as a verification-oriented component-based system specification. In *Proceedings of SAVCBS'05*, pages 31–38. ACM, 2005.

[5] S. Chaki, E. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. Concurrent software verification with states, events, and deadlocks. *Formal Aspects of Computing*, 17(4):461–483, 2005.

[6] S. Chaki, E. M. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. State/event-based software model checking. In *Proceedings of IFM'04*, volume 2999 of *LNCS*, pages 128–147. Springer-Verlag, 2004.

[7] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. Cambridge, London, MIT Press, 1999.

[8] CoIn Tool – Formal Verification Tool for Component Interaction Automata. URL http://anna.fi.muni.cz/coin/tool/.

[9] L. de Alfaro and T. A. Henzinger. Interface-based design. In *Proceedings of the 2004 Marktoberdorf Summer School*. Kluwer, The Netherlands, 2005.

[10] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of PSTV'95*, pages 3–18, Warsaw, Poland, 1995. Chapman & Hall.

[11] P. Godefroid. Using partial orders to improve automatic verification methods. In *Proceedings of CAV'90*, volume 531 of *LNCS*, pages 176–185. Springer-Verlag, 1991.

[12] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[13] N. A. Lynch and M. R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, 1989.

[14] D. Peled. All from one, one from all: on model checking using representatives. In *Proceedings of CAV'93*, volume 697 of *LNCS*, pages 409–423. Springer-Verlag, 1993.

[15] D. Peled. Combining partial order reductions with on-the-fly model-checking. In *Proceedings of CAV'94*, volume 818 of *LNCS*, pages 377–390. Springer-Verlag, 1994.

[16] D. Peled and T. Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Information Processing Letters*, 63(5):243–246, 1997.

[17] A. Valmari. A stubborn attack on state explosion. In *Proceedings of CAV'90*, volume 531 of *LNCS*, pages 156–165. Springer-Verlag, 1990.

[18] B. Zimmerova, P. Vařeková, N. Beneš, I. Černá, L. Brim, and J. Sochor. *The Common Component Modeling Example: Comparing Software Component Models*, chapter Component-Interaction Automata Approach (CoIn). To appear in LNCS, 2008.