The Inconsistent Labelling Problem of Stutter-Preserving Partial-Order Reduction

Thomas Neele[1(✉)], Antti Valmari[2], and Tim A.C. Willemse[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{t.s.neele, t.a.c.willemse}@tue.nl
[2] University of Jyväskylä, Jyväskylä, Finland
antti.valmari@jyu.fi

Abstract. In model checking, partial-order reduction (POR) is an effective technique to reduce the size of the state space. Stubborn sets are an established variant of POR and have seen many applications over the past 31 years. One of the early works on stubborn sets shows that a combination of several conditions on the reduction is sufficient to preserve stutter-trace equivalence, making stubborn sets suitable for model checking of linear-time properties. In this paper, we identify a flaw in the reasoning and show with a counter-example that stutter-trace equivalence is not necessarily preserved. We propose a solution together with an updated correctness proof. Furthermore, we analyse in which formalisms this problem may occur. The impact on practical implementations is limited, since they all compute a correct approximation of the theory.

1 Introduction

In formal methods, model checking is a technique to automatically decide the correctness of a system's design. The many interleavings of concurrent processes can cause the state space to grow exponentially with the number of components, known as the *state-space explosion* problem. *Partial-order reduction* (POR) is one technique that can alleviate this problem. Several variants of POR exist, such as *ample sets* [11], *persistent set* [7] and *stubborn sets* [16,21]. For each of those variants, sufficient conditions for preservation of stutter-trace equivalence have been identified. Since LTL without the next operator (LTL$_{-X}$) is invariant under finite stuttering, this allows one to check most LTL properties under POR.

However, the correctness proofs for these methods are intricate and not reproduced often. For stubborn sets, LTL$_{-X}$-preserving conditions and an accompanying correctness result were first presented in [15], and discussed in more detail in [17]. While trying to reproduce the proof for [17, Theorem 2] (see also Theorem 1 in the current work), we ran into an issue while trying to prove a certain property of the construction used in the original proof [17, Construction 1]. This led us to discover that stutter-trace equivalence is not necessarily preserved. We will refer to this as the *inconsistent labelling problem.* The essence of the problem is that POR in general, and the proofs in [17] in particular, reason mostly about actions, which label the transitions. The only relevance of the state labelling is that it determines which actions are *visible*. On the other hand, stutter-trace equivalence and the LTL semantics are purely based on state labels. The correctness proof in [17] does not deal properly with this disparity. Further investigation shows that the same problem also occurs in two works of Beneš *et al.* [2,3], who apply ample sets to state/event LTL model checking.

Consequently, any application of stubborn sets in LTL$_{-X}$ model checking is possibly unsound, both for safety and liveness properties. In literature, the correctness of several theories [9,10,18] relies on the incorrect theorem.

Our contributions are as follows:

– We prove the existence of the inconsistent labelling problem with a counter-example. This counter-example is valid for weak stubborn sets and, with a small modification, in a non-deterministic setting for strong stubborn sets.
– We propose to strengthen one of the stubborn set conditions and show that this modification resolves the issue (Theorem 2).
– We analyse in which circumstances the inconsistent labelling problem occurs and, based on the conclusions, discuss its impact on existing literature. This includes a thorough analysis of Petri nets and several different notions of invisible transitions and atomic propositions.

Our investigation shows that probably all practical implementations of stubborn sets compute an approximation which resolves the inconsistent labelling problem. Furthermore, POR methods based on the standard independence relation, such as ample sets and persistent sets, are not affected.

The rest of the paper is structured as follows. In Section 2, we introduce the basic concepts of stubborn sets and stutter-trace equivalence, which is not preserved in the counter-example of Section 3. A solution to the inconsistent labelling problem is discussed in Section 4, together with an updated correctness proof. Sections 5 and 6 discuss several settings in which correctness is not affected. Finally, Section 7 presents related work and Section 8 presents a conclusion.

2 Preliminaries

Since LTL relies on state labels and POR relies on edge labels, we assume the existence of some fixed set of atomic propositions AP to label the states and a fixed set of edge labels Act, which we will call *actions*. Actions are typically denoted with the letter a.

Definition 1. *A* labelled state transition system*, short* LSTS*, is a directed graph* $TS = (S, \rightarrow, \hat{s}, L)$*, where:*

– S *is the state space;*
– $\rightarrow \subseteq S \times Act \times S$ *is the transition relation;*
– $\hat{s} \in S$ *is the initial state; and*
– $L : S \rightarrow 2^{AP}$ *is a function that labels states with atomic propositions.*

We write $s \xrightarrow{a} t$ whenever $(s, a, t) \in \rightarrow$. A *path* is a (finite or infinite) alternating sequence of states and actions: $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots$. We sometimes omit the intermediate and/or final states if they are clear from the context or not relevant, and write $s \xrightarrow{a_1 \dots a_n} t$ or $s \xrightarrow{a_1 \dots a_n}$ for finite paths and $s \xrightarrow{a_1 a_2 \dots}$ for infinite paths. Paths that start in the initial state $\hat{s}$ are called *initial paths.* Given a path $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \dots$, the *trace* of π is the sequence of state labels observed along π, *viz.* $L(s_0)L(s_1)L(s_2)\dots$. An action a is *enabled* in a state s, notation $s \xrightarrow{a}$, if and only if there is a transition $s \xrightarrow{a} t$ for some t. In a given LSTS TS, $\mathit{enabled}_{TS}(s)$ is the set of all enabled actions in a state s. A set $\mathcal{I}$ of *invisible* actions is chosen such that if (but not necessarily only if) $a \in \mathcal{I}$, then for all states s and t, $s \xrightarrow{a} t$ implies $L(s) = L(t)$. Note that this definition allows the set $\mathcal{I}$ to be under-approximated. An action that is not invisible is called *visible.* We say TS is *deterministic* if and only if $s \xrightarrow{a} t$ and $s \xrightarrow{a} t'$ imply $t = t'$, for all states s, t and t' and actions a. To indicate that TS is not necessarily deterministic, we say TS is *non-deterministic.*

2.1 Stubborn sets

In POR, *reduction functions* play a central role. A reduction function $r : S \to 2^{Act}$ indicates which transitions to explore in each state. When starting at the initial state $\hat{s}$, a reduction function induces a *reduced LSTS* as follows.

Definition 2. *Let $TS = (S, \rightarrow, \hat{s}, L)$ be an LSTS and $r : S \to 2^{Act}$ a reduction function. Then the* reduced LSTS *induced by r is defined as $TS_r = (S_r, \rightarrow_r, \hat{s}, L_r)$, where L_r is the restriction of L on S_r, and S_r and $\rightarrow_r$ are the smallest sets such that the following holds:*

- *$\hat{s} \in S_r$; and*
- *If $s \in S_r$, $s \xrightarrow{a} t$ and $a \in r(s)$, then $t \in S_r$ and $s \xrightarrow{a}_r t$.*

Note that we have $\rightarrow_r \subseteq \rightarrow$. In the remainder of this paper, we will assume the reduced LSTS is finite. This is essential for the correctness of the approach detailed below. In general, a reduction function is not guaranteed to preserve almost any property of an LSTS. Below, we list a number of conditions that have been proposed in literature; they aim to preserve LTL$_{-X}$. Here, we call an action a a *key action* in s iff for all paths $s \xrightarrow{a_1 \dots a_n} s'$ such that $a_1 \notin r(s), \dots, a_n \notin r(s)$, it holds that $s' \xrightarrow{a}$. We typically denote key actions by a_{key}.

D0 If $\mathit{enabled}(s) \neq \emptyset$, then $r(s) \cap \mathit{enabled}(s) \neq \emptyset$.

D1 For all $a \in r(s)$ and $a_1 \notin r(s), \dots, a_n \notin r(s)$, if $s \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s'_n$, then there are states $s', s'_1, \dots, s'_{n-1}$ such that $s \xrightarrow{a} s' \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'_n$.

D2 Every enabled action in $r(s)$ is a key action in s.

D2w If $\mathit{enabled}(s) \neq \emptyset$, then $r(s)$ contains a key action in s.

V If $r(s)$ contains an enabled visible action, then it contains all visible actions.

I If an invisible action is enabled, then $r(s)$ contains an invisible key action.

L For every visible action a, every cycle in the reduced LSTS contains a state s such that $a \in r(s)$.

$$\begin{tikzcd}
s \arrow[r, "a_1"] & s_1 \arrow[r, dash] & \cdots \arrow[r] & s_{n-1} \arrow[r, "a_n"] & s_n \arrow[d, "a"] & & s \arrow[r, "a_1"] \arrow[d, "a"] & s_1 \arrow[r, dash] & \cdots \arrow[r] & s_{n-1} \arrow[r, "a_n"] & s_n \arrow[d, "a"] \\
 & & & & s'_n & \Rightarrow & s' \arrow[r, "a_1"] & s'_1 \arrow[r, dash] & \cdots \arrow[r] & s'_{n-1} \arrow[r, "a_n"] & s'_n
\end{tikzcd}$$

Fig. 1: Visual representation of condition **D1**.

These conditions are used to define *strong* and *weak* stubborn sets in the following way.

Definition 3. *A reduction function $r : S \rightarrow 2^{Act}$ is a* strong stubborn set *iff for all states $s \in S$, the conditions **D0**, **D1**, **D2**, **V**, **I**, **L** all hold.*

Definition 4. *A reduction function $r : S \rightarrow 2^{Act}$ is a* weak stubborn set *iff for all states $s \in S$, the conditions **D1**, **D2w**, **V**, **I**, **L** all hold.*

Below, we also use 'weak/strong stubborn set' to refer to the set of actions $r(s)$ in some state s. First, note that key actions are always enabled, by setting $n = 0$. Furthermore, a stubborn set can never introduce new deadlocks, either by **D0** or **D2w**. Condition **D1** enforces that a key action $a_{\mathsf{key}} \in r(s)$ does not disable other paths that are not selected for the stubborn set. A visual representation of condition **D1** can be found in Figure 1. When combined, **D1** and **D2w** are sufficient conditions for preservation of deadlocks. Condition **V** enforces that the paths $s \xrightarrow{a_1 \dots a_n a} s'_n$ and $s \xrightarrow{a a_1 \dots a_n} s'_n$ in **D1** contain the same sequence of visible actions. The purpose of condition **I** is to preserve the possibility to perform an invisible action, if one is enabled. Finally, we have condition **L** to deal with the *action-ignoring problem*, which occurs when an action is never selected for the stubborn set and always ignored. Since we assume that the reduced LSTS is finite, it suffices to reason in **L** about every cycle instead of every infinite path. The combination of **I** and **L** helps to preserve divergences (infinite paths containing only invisible actions).

Conditions **D0** and **D2** together imply **D2w**, and thus every strong stubborn set is also a weak stubborn set. Since the reverse does not necessarily hold, weak stubborn sets might offer more reduction.

2.2 Weak and Stutter Equivalence

To reason about the similarity of an LSTS TS and its reduced LSTS TS_r, we introduce the notions of *weak equivalence*, which operates on actions, and *stutter equivalence*, which operates on states. The definitions are generic, so that they can also be used in Section 6.

Definition 5. *Two paths π and π' are weakly equivalent with respect to a set of actions A, notation $\pi \sim_A \pi'$, if and only if they are both finite or both infinite and their respective projections on $Act \setminus A$ are equal.*

Definition 6. *The* no-stutter trace *under labelling L of a path $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots$ is the sequence of those $L(s_i)$ such that $i = 0$ or $L(s_i) \neq L(s_{i-1})$. Paths π and π' are stutter equivalent under L, notation $\pi \triangleq_L \pi'$, iff they are both finite or both infinite, and they yield the same no-stutter trace under L.*

We typically consider weak equivalence with respect to the set of invisible actions $\mathcal{I}$. In that case, we write $\pi \sim \pi'$. We also omit the subscript for stutter equivalence when reasoning about the standard labelling function and write $\pi \triangleq \pi'$. Remark that stutter equivalence is invariant under finite repetitions of state labels, hence its name. We lift both equivalences to LSTSs, and say that TS and TS' are *weak-trace equivalent* iff for every initial path π in TS, there is a weakly equivalent initial path π' in TS' and vice versa. Likewise, TS and TS' are *stutter-trace equivalent* iff for every initial path π in TS, there is a stutter equivalent initial path π' in TS' and vice versa.

In general, weak equivalence and stutter equivalence are incomparable, even for initial paths. However, for some LSTSs, these notions can be related in a certain way. We formalise this in the following definition.

Definition 7. *Let TS be an LSTS and π and π' two paths in TS that both start in some state s. Then, TS is* labelled consistently *iff $\pi \sim \pi'$ implies $\pi \triangleq \pi'$.*

Note that if an LSTS is labelled consistently, then in particular all weakly equivalent initial paths are also stutter equivalent. Hence, if an LSTS TS is labelled consistently and weak-trace equivalent to a subgraph TS', then TS and TS' are also stutter-trace equivalent.

Stubborn sets as defined in the previous section aim to preserve stutter-trace equivalence between the original and the reduced LSTS. The motivation behind this is that two stutter-trace equivalent LSTSs satisfy exactly the same formulae [1] in LTL$_{-X}$. The following theorem, which is frequently cited in literature [9,10,18], aims to show that stubborn sets indeed preserve stutter-trace equivalence. Its original formulation reasons about the validity of an arbitrary LTL$_{-X}$ formula. Here, we give the alternative formulation based on stutter-trace equivalence.

Theorem 1. *[17, Theorem 2] Given an LSTS TS and a weak/strong stubborn set r, then the reduced LSTS TS_r is stutter-trace equivalent to TS.*

The original proof correctly concludes that the stubborn set method preserves the order of visible actions in the reduced LSTS, *i.e.*, $TS \sim TS_r$. However, this only implies preservation of stutter-trace equivalence ($TS \triangleq TS_r$) if the full LSTS is labelled consistently, so Theorem 1 is invalid in the general case. In the next section, we will see a counter-example which exploits this fact.

3 Counter-Example

Consider the LSTS in Figure 2, which we will refer to as TS^C. There is only one atomic proposition q, which holds in the grey states and is false in the other states. The initial state $\hat{s}$ is marked with an incoming arrow. First, note that this LSTS is deterministic. The actions a_1, a_2 and a_3 are visible and a and a_{key} are invisible. By setting $r(\hat{s}) = \{a, a_{\mathsf{key}}\}$, which is a weak stubborn set, we obtain a reduced LSTS TS_r^C that does not contain the dashed states and transitions. The original LSTS contains the trace $\emptyset\{q\}\emptyset\emptyset\{q\}^\omega$, obtained by following the path with actions $a_1 a_2 a a_3^\omega$. However, the reduced LSTS does not contain a stutter equivalent trace. This is also witnessed by the LTL$_{-X}$ formula $\Box(q \Rightarrow \Box(q \vee \Box\neg q))$, which holds for TS_r^C, but not for TS^C.

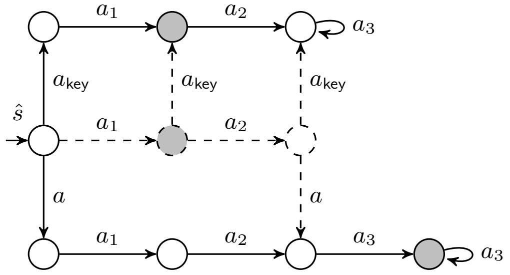


Fig. 2: Counter-example showing that stubborn sets do not preserve stutter-trace equivalence. Grey states are labelled with $\{q\}$. The dashed transitions and states are not present in the reduced LSTS.

A very similar example can be used to show that strong stubborn sets suffer from the same problem. Consider again the LSTS in Figure 2, but assume that $a = a_{\mathsf{key}}$, making the LSTS non-deterministic. Now, $r(\hat{s}) = \{a\}$ is a strong stubborn set and again the trace $\emptyset\{q\}\emptyset\emptyset\{q\}^\omega$ is not preserved in the reduced LSTS. In Section 4.3, we will see why the inconsistent labelling problem does not occur for deterministic systems under strong stubborn sets.

The core of the problem lies in the fact that condition **D1**, even when combined with **V**, does not enforce that the two paths it considers are stutter equivalent. Consider the paths $s \xrightarrow{a}$ and $s \xrightarrow{a_1 a_2 a}$ and assume that $a \in r(s)$ and $a_1 \notin r(s), a_2 \notin r(s)$. Condition **V** ensures that at least one of the following two holds: (i) a is invisible, or (ii) a_1 and a_2 are invisible. Half of the possible scenarios are depicted in Figure 3; the other half are symmetric. Again, the grey states (and only those states) are labelled with $\{q\}$.

The two cases delimited with a solid line are problematic. In both LSTSs, the paths $s \xrightarrow{a_1 a_2 a} s'$ and $s \xrightarrow{a a_1 a_2} s'$ are weakly equivalent, since a is invisible. However, they are not stutter equivalent, and therefore these LSTSs are not labelled consistently. The topmost of these two LSTSs forms the core of the counter-example TS^C, with the rest of TS^C serving to satisfy condition **D2**/**D2w**.

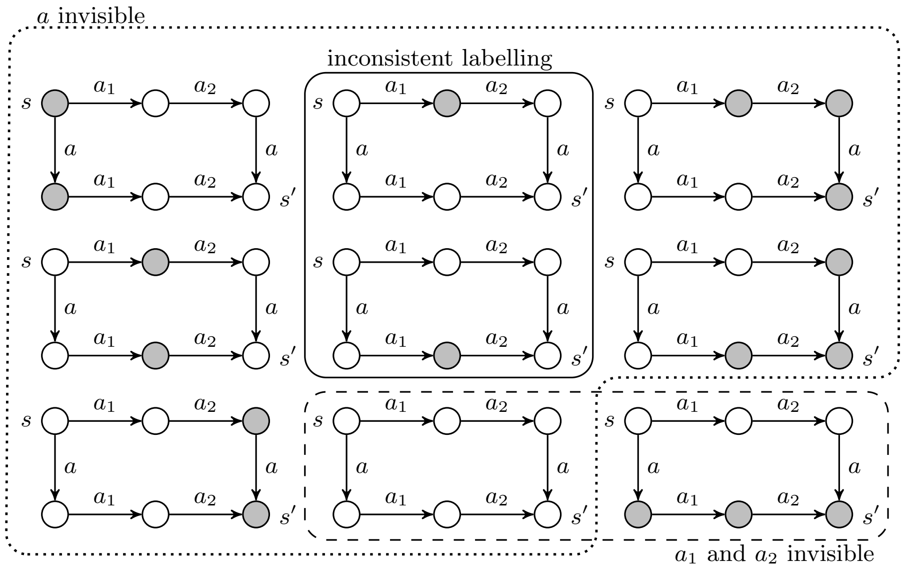


Fig. 3: Nine possible scenarios when $a \in r(s)$ and $a_1 \notin r(s), a_2 \notin r(s)$, according to conditions **D1** and **V**. The dotted and dashed lines indicate when a or a_1, a_2 are invisible, respectively.

4 Strengthening Condition D1

To fix the issue with inconsistent labelling, we propose to strengthen condition **D1** as follows.

D1' For all $a \in r(s)$ and $a_1 \notin r(s), \dots, a_n \notin r(s)$, if $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s'_n$, then there are states $s', s'_1, \dots, s'_{n-1}$ such that $s \xrightarrow{a} s' \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'_n$. Furthermore, if a is invisible, then $s_i \xrightarrow{a} s'_i$ for every $1 \leq i < n$.

This new condition **D1'** provides a form of *local* consistent labelling when one of $a_1, \dots, a_n$ is visible. In this case, **V** implies that a is invisible and, consequently, the presence of transitions $s_i \xrightarrow{a} s'_i$ implies $L(s_i) = L(s'_i)$. Hence, the problematic cases of Figure 3 are resolved; a correctness proof is given below.

Condition **D1'** is very similar to condition **C1** [5], which is common in the context of ample sets. However, **C1** requires that action a is *globally* independent of each of the actions $a_1, \dots, a_n$, while **D1'** merely requires a kind of *local* independence. Persistent sets [7] also rely on a condition similar to **D1'**, and require local independence.

4.1 Implementation

In practice, most, if not all, implementations of stubborn sets approximate **D1** based on a binary relation $\leadsto_s$ on actions. This relation may (partly) depend on the current state s and it is defined such that **D1** can be satisfied by ensuring that if $a \in r(s)$ and $a \rightsquigarrow_s a'$, then also $a' \in r(s)$. A set satisfying **D0**, **D1**, **D2**, **D2w**, **V** and/or **I** can be found by searching for a suitable *strongly connected component* in the graph $(Act, \rightsquigarrow_s)$. Condition **L** is dealt with by other techniques.

Practical implementations construct $\rightsquigarrow_s$ by analysing how any two actions a and a' interact. If a is enabled, the simplest (but not necessarily the best possible) strategy is to make $a \rightsquigarrow_s a'$ if and only if a and a' access at least one variable in common. This can be relaxed, for instance, by not considering commutative accesses, such as writing to and reading from a FIFO buffer. As a result, $\rightsquigarrow_s$ can only detect reduction opportunities in (sub)graphs of the shape

$$\begin{tikzcd}
s \arrow[r, "a_1"] \arrow[d, "a"] & s_1 \arrow[r, dashed, "\cdots"] \arrow[d, "a"] & s_{n-1} \arrow[r, "a_n"] \arrow[d, "a"] & s_n \arrow[d, "a"] \\
s' \arrow[r, "a_1"] & s'_1 \arrow[r, dashed, "\cdots"] & s'_{n-1} \arrow[r, "a_n"] & s'_n
\end{tikzcd}$$

where $a \in r(s)$ and $a_1 \notin r(s), \ldots, a_n \notin r(s)$. The presence of the vertical a transitions in $s_1, \ldots, s_{n-1}$ implies that **D1'** is also satisfied by such implementations.

4.2 Correctness

To show that **D1'** indeed resolves the inconsistent labelling problem, we reproduce the construction in the original proof [17, Construction 1] in two lemmata and show that it preserves stutter equivalence. Below, recall that $\rightarrow_r$ indicates which transitions occur in the reduced state space.

Lemma 1. *Let r be a weak stubborn set, where condition **D1** is replaced by **D1'**, and $\pi = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n \xrightarrow{a} s'_n$ a path such that $a_1 \notin r(s_0), \ldots, a_n \notin r(s_0)$ and $a \in r(s_0)$. Then, there is a path $\pi' = s_0 \xrightarrow{a}_r s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'_n$ such that $\pi \triangleq \pi'$.*

Proof. The existence of π' follows directly from condition **D1'**. Due to condition **V** and our assumption that $a_1 \notin r(s_0), \ldots, a_n \notin r(s_0)$, it cannot be the case that a is visible and at least one of $a_1, \ldots, a_n$ is visible. If a is invisible, then the traces of $s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n$ and $s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'_n$ are equivalent, since **D1'** implies that $s_i \xrightarrow{a} s'_i$ for every $0 \leq i \leq n$, so $L(s'_i) = L(s_i)$. Otherwise, if all of $a_1, \ldots, a_n$ are invisible, then the sequences of labels observed along π and π' have the shape $L(s_0)^{n+1} L(s'_0)$ and $L(s_0) L(s'_0)^{n+1}$, respectively. We conclude that $\pi \triangleq \pi'$. □

Lemma 2. *Let r be a weak stubborn set, where condition **D1** is replaced by **D1'**, and $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ a path such that $a_i \notin r(s_0)$ for any a_i that occurs in π. Then, the following holds:*

- *If π is of finite length $n > 0$, there exist an action a_{key}, a state s'_n such that $s_n \xrightarrow{a_{\mathsf{key}}} s'_n$ and a path $\pi' = s_0 \xrightarrow{a_{\mathsf{key}}}_r s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'_n$.*
- *If π is infinite, there exists a path $\pi' = s_0 \xrightarrow{a_{\mathsf{key}}}_r s'_0 \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \ldots$ for some action a_{key}.*

In either case, $\pi \triangleq \pi'$.

Proof. Let K be the set of key actions in s. If a_1 is invisible, K contains at least one invisible action, due to **I**. Otherwise, if a_1 is visible, we reason that K is not empty (condition **D2w**) and all actions in $r(s_0)$, and thus also all actions in K, are invisible, due to **V**. In the remainder, let a_{key} be an invisible key action.

In case π has finite length n, the existence of $s_n \xrightarrow{a_{\mathsf{key}}} s'_n$ and $s_0 \xrightarrow{a_{\mathsf{key}}}_r s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'_n$ follows from the definition of key actions and **D1'**, respectively.

If π is infinite, we can apply the definition of key actions and **D1'** successively to obtain a path $\pi_i = s_0 \xrightarrow{a_{\mathsf{key}}} s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_i} s'_i$ for every $i \geq 0$, with $s_j \xrightarrow{a_{\mathsf{key}}} s'_j$ for every $1 \leq j < i$. Since the reduced state space is finite, infinitely many of these paths must use the same state as s'_0. At most one of them ends at s'_0 (the one with $i = 0$), so infinitely many continue from s'_0. Of them, infinitely many must use the same s'_1, again because the reduced state space is finite. Again, at most one of them is lost because of ending at s'_1. This reasoning can continue without limit, proving the existence of $\pi' = s_0 \xrightarrow{a_{\mathsf{key}}}_r s'_0 \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \ldots$, with $s_j \xrightarrow{a_{\mathsf{key}}} s'_j$ for every $j \geq 0$.

Since a_{key} is invisible, we have $L(s_j) = L(s'_j)$ for every $j \geq 0$. This implies $\pi \triangleq \pi'$. □

Lemmata 1 and 2 coincide with branches 1 and 2 of [17, Construction 1], respectively, but contain the stronger result that $\pi \triangleq \pi'$. Thus, when applied in the proof of [17, Theorem 2] (see also Theorem 1), this yields the result that stubborn sets with condition **D1'** preserve stutter-trace equivalence.

Theorem 2. *Given an LSTS TS and weak/strong stubborn set r, where condition* ***D1*** *is replaced by* ***D1'****, then the reduced LSTS TS_r is stutter-trace equivalent to TS.*

We do not reproduce the complete proof, but provide insight into the application of the lemmata with the following example.

Example 1. Consider the path obtained by following $a_1a_2a_3$ in Figure 4. Lemmata 1 and 2 show that $a_1a_2a_3$ can always be mimicked in the reduced LSTS, while preserving stutter equivalence. In this case, the path is mimicked by the path corresponding to $a_{\mathsf{key}}a_2a_1a'_{\mathsf{key}}a_3$, drawn with dashes. The new path reorders the actions a_1, a_2 and a_3 according to the construction of Lemma 1 and introduces the key actions a_{key} and a'_{key} according to Lemma 2. □

We remark that Lemma 2 also holds if the reduced LSTS is infinite, but finitely branching.

4.3 Deterministic LSTSs

As already noted in Section 3, strong stubborn sets for deterministic systems do not suffer from the inconsistent labelling problem. The following lemma, which also appeared as [20, Lemma 4.2], shows why.

Lemma 3. *For deterministic LSTSs, conditions* ***D1*** *and* ***D2*** *together imply* ***D1'****.*

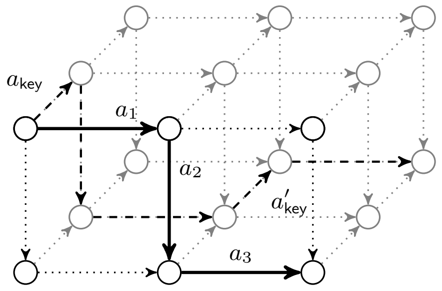


Fig. 4: Example of how the trace a_1, a_2, a_3 can be mimicked by introducing additional actions and moving a_2 to the front (dashed trace). Transitions that are drawn in parallel have the same label.

5 Safe Logics

In this section, we will identify two logics, *viz.* reachability and CTL$_{-X}$, which are not affected by the inconsistent labelling problem. This is either due to their limited expressivity or the extra POR conditions that are required.

5.1 Reachability properties

Although the counter-example of Section 3 shows that stutter-trace equivalence is in general not preserved by stubborn sets, some fragments of LTL$_{-X}$ are preserved. One such class of properties is reachability properties, which are of the shape $\Box f$ or $\Diamond f$, where f is a formula not containing temporal operators.

Theorem 3. *Let TS be an LSTS, r a reduction function that satisfies either* ***D0****,* ***D1****,* ***D2****,* ***V*** *and* ***L*** *or* ***D1****,* ***D2w****,* ***V*** *and* ***L*** *and TS_r the reduced LSTS. For all possible labellings $l \subseteq AP$, TS contains a path to a state s such that $L(s) = l$ iff TS_r contains a path to a state s' such that $L(s') = l$.*

Proof. The 'if' case is trivial, since TS_r is a subgraph of TS. For the 'only if' case, we reason as follows. Let $TS = (S, \rightarrow, \hat{s}, L)$ be an LSTS and $\pi = s_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} s_n$ a path such that $s_0 = \hat{s}$. We mimic this path by repeatedly taking some enabled action a that is in the stubborn set, according to the following schema. Below, we assume the path to be mimicked contains at least one visible action. Otherwise, its first state would have the same labelling as s_n.

1. If there is an i such that $a_i \in r(s_0)$, we consider the smallest such i, *i.e.*, $a_1 \notin r(s_0), \ldots, a_{i-1} \notin r(s_0)$. Then, we can shift a_i forward by **D1**, move towards s_n along $s_0 \xrightarrow{a_i} s'_0$ and continue by mimicking $s'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_{i-1}} s_i \xrightarrow{a_{i+1}} \cdots \xrightarrow{a_n} s_n$.
2. If all of $a_1 \notin r(s_0), \ldots, a_n \notin r(s_0)$, then, by **D0** and **D2** or by **D2w**, there is a key action a_{key} in s_0. By the definition of key actions and **D1**, a_{key} leads to a state s'_0 from which we can continue mimicking the path $s'_0 \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'_n$. Note that $L(s_n) = L(s'_n)$, since a_{key} is invisible by condition **V**.

The second case cannot be repeated infinitely often, due to condition **L**. Hence, after a finite number of steps, we reach a state s'_n with $L(s'_n) = L(s_n)$. $\square$

We remark that more efficient mechanisms for reachability checking under POR have been proposed, such as condition **S** [21], which can replace **L**, or conditions based on *up-sets* [13]. Another observation is that model checking of LTL$_{-X}$ properties can be reduced to reachability checking by computing the cross-product of a Büchi automaton and an LSTS [1], in the process resolving the inconsistent labelling problem. Peled [12] shows how this approach can be combined with POR, but please see [14].

5.2 Deterministic LSTSs and CTL$_{-X}$ Model Checking

In this section, we will consider the inconsistent labelling problem in the setting of CTL$_{-X}$ model checking. When applying stubborn sets in that context, stronger conditions are required to preserve the branching structure that CTL$_{-X}$ reasons about. Namely, the original LSTS must be deterministic and one more condition needs to be added [5]:

C4 Either $r(s) = Act$ or $r(s) \cap enabled(s) = \{a\}$ for some $a \in Act$.

We slightly changed its original formulation to match the setting of stubborn sets. A weaker condition, called **Ä8**, which does not require determinism of the whole LSTS is proposed in [19]. With **C4**, strong and weak stubborn sets collapse, as shown by the following lemma.

Lemma 4. *Conditions **D2w** and **C4** together imply **D0** and **D2**.*

Proof. Let TS be an LSTS, s a state and r a reduction function that satisfies **D2w** and **C4**. Condition **D0** is trivially implied by **C4**. Using **C4**, we distinguish two cases: either $r(s)$ contains precisely one enabled action a, or $r(s) = Act$. In the former case, this single action a must be a key action, according to **D2w**. Hence, **D2**, which requires that all enabled actions in $r(s)$ are key actions, is satisfied. Otherwise, if $r(s) = Act$, we consider an arbitrary action a that satisfies **D2**'s precondition that $s \xrightarrow{a}$. Given a path $s \xrightarrow{a_1 \dots a_n}$, the condition that $a_1 \notin r(s), \dots, a_n \notin r(s)$ only holds if $n = 0$. We conclude that **D2**'s condition $s \xrightarrow{a_1 \dots a_n a}$ is satisfied by the assumption $s \xrightarrow{a}$. $\square$

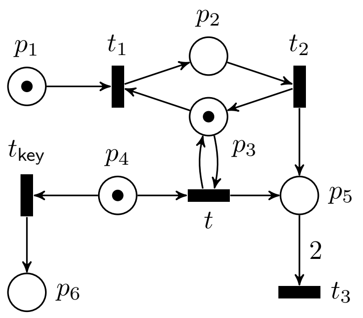
It follows from Lemmata 3 and 4 and Theorem 2 that CTL$_{-X}$ model checking of deterministic systems with stubborn sets does not suffer from the inconsistent labelling problem. The same holds for condition **Ä8**, as already shown in [19].

6 Petri Nets

Petri nets are a widely-known formalism for modelling concurrent processes and have seen frequent use in the application of stubborn-set theory [4,10,21,22]. A Petri net contains a set of *places* P and a set of *structural transitions* T.

Arcs between places and structural transitions are weighted according to a total function $W : (P{\times}T)\cup(T{\times}P) \to \mathbb{N}$. The state space of the underlying LSTS is the set $\mathcal{M}$ of all *markings*; a marking m is a function $P \to \mathbb{N}$, which assigns a number of *tokens* to each place. The LSTS contains a transition $m \xrightarrow{t} m'$ iff $m(p) \geq W(p,t)$ and $m'(p) = m(p) - W(p,t) + W(t,p)$ for all places $p \in P$. As before, we assume the LSTS contains some labelling function $L : \mathcal{M} \to 2^{AP}$. More details on the labels are given below. Note that markings and structural transitions take over the role of states and actions respectively. The set of markings reachable under $\to$ from some *initial marking* $\hat{m}$ is denoted $\mathcal{M}_{reach}$.

Example 2. Consider the Petri net with initial marking $\hat{m}$ below on the left. Here, all arcs are weighted 1, except for the arc from p_5 to t_2, which is weighted 2. Its LSTS is infinite, but the reachable substructure is depicted on the right. The number of tokens in each of the places $p_1, \ldots, p_6$ is inscribed in the nodes, the state labels (if any) are written beside the nodes.



$$\begin{tikzcd}
 & & \{q_p\} & & \\
 & 101001 \arrow[r, "t_1"] & 010001 \arrow[r, "t_2"] & 001011 & \\
{} \arrow[r, "\hat{m}"] & 101100 \arrow[u, "t_{\mathsf{key}}"] \arrow[r, "t_1"] \arrow[dd, "t"] & 010100 \arrow[u, "t_{\mathsf{key}}"] \arrow[r, "t_2"] & 001110 \arrow[u, "t_{\mathsf{key}}"] \arrow[dd, "t"] & \\
 & & \{q_p\} & & \\
 & 101010 \arrow[r, "t_1"] & 010010 \arrow[r, "t_2"] & 001020 \arrow[r, "t_3"] & 001000 \\
 & & \{q_l\} & & \{q\}
\end{tikzcd}$$

The LSTS practically coincides with the counter-example of Section 3. Only the self-loops are missing and the state labelling, with atomic propositions q, q_p and q_l, differs slightly; the latter will be explained later. For now, note that t and t_{key} are invisible and that the trace $\emptyset\{q_p\}\emptyset\emptyset\{q\}$, which occurs when firing transitions $t_1 t_2 t t_3$ from $\hat{m}$, can be lost when reducing with weak stubborn sets. □

In the remainder of this section, we fix a Petri net $(P, T, W, \hat{m})$ and its LSTS $(\mathcal{M}, \to, \hat{m}, L)$. Below, we consider three different types of atomic propositions. Firstly, polynomial propositions [4] are of the shape $f(p_1, \ldots, p_n) \bowtie k$ where f is a polynomial over $p_1, \ldots, p_n$, $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$ and $k \in \mathbb{Z}$. Such a proposition holds in a marking m iff $f(m(p_1), \ldots, m(p_n)) \bowtie k$. A linear proposition [10] is similar, but the function f over places must be linear and $f(0, \ldots, 0) = 0$, *i.e.*, linear propositions are of the shape $k_1 p_1 + \cdots + k_n p_n \bowtie k$, where $k_1, \ldots, k_n, k \in \mathbb{Z}$. Finally, we have arbitrary propositions [22], whose shape is not restricted and which can hold in any given set of markings.

Several other types of atomic propositions can be encoded as polynomial propositions. For example, *fireable*(t) [4,10], which holds in a marking m iff t is enabled in m, can be encoded as $\prod_{p \in P} \prod_{i=0}^{W(p,t)-1} (p - i) \geq 1$. The proposition *deadlock*, which holds in markings where no structural transition is enabled, does not require special treatment in the context of POR, since it is already preserved by **D1** and **D2w**. The sets containing all linear and polynomial propositions are henceforward called AP_l and AP_p, respectively. The corresponding labelling functions are defined as $L_l(m) = L(m) \cap AP_l$ and $L_p(m) = L(m) \cap AP_p$ for all markings m. Below, the two stutter equivalences $\triangleq_{L_l}$ and $\triangleq_{L_p}$ that follow from the new labelling functions are abbreviated $\triangleq_l$ and $\triangleq_p$, respectively. Note that $AP \supseteq AP_p \supseteq AP_l$ and $\triangleq \subseteq \triangleq_p \subseteq \triangleq_l$.

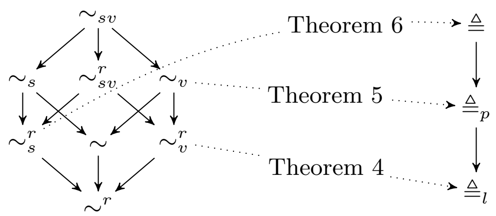
For the purpose of introducing several variants of invisibility, we reformulate and generalise the definition of invisibility from Section 2. Given an atomic proposition $q \in AP$, a relation $\mathcal{R} \subseteq \mathcal{M} \times \mathcal{M}$ is *q-invisible* if and only if $(m, m') \in \mathcal{R}$ implies $q \in L(m) \Leftrightarrow q \in L(m')$. We consider a structural transition t q-invisible iff its corresponding relation $\{(m, m') \mid m \xrightarrow{t} m'\}$ is q-invisible. Invisibility is also lifted to sets of atomic propositions: given a set $AP' \subseteq AP$, relation $\mathcal{R}$ is *AP'-invisible* iff it is q-invisible for all $q \in AP'$. If $\mathcal{R}$ is AP-invisible, we plainly say that $\mathcal{R}$ is *invisible*. AP'-invisibility and invisibility carry over to structural transitions. We sometimes refer to invisibility as *ordinary invisibility* for emphasis. Note that the set of invisible structural transitions $\mathcal{I}$ is no longer an under-approximation, but contains exactly those structural transitions t for which $m \xrightarrow{t} m'$ implies $L(m) = L(m')$ (cf. Section 2).

We are now ready to introduce three orthogonal variations on invisibility. Firstly, relation $\mathcal{R} \subseteq \mathcal{M} \times \mathcal{M}$ is *reach q-invisible* [21] iff $\mathcal{R} \cap (\mathcal{M}_{reach} \times \mathcal{M}_{reach})$ is q-invisible, *i.e.*, all the pairs of reachable markings $(m, m') \in \mathcal{R}$ agree on the labelling of q. Secondly, $\mathcal{R}$ is *value q-invisible* if (i) q is polynomial and for all $(m, m') \in \mathcal{R}$, $f(m(p_1), \dots, m(p_n)) = f(m'(p_1), \dots, m'(p_n))$; or if (ii) q is not polynomial and $\mathcal{R}$ is q-invisible. Intuitively, this means that the value of polynomial f never changes between two markings $(m, m') \in \mathcal{R}$. Reach and value invisibility are lifted to structural transitions and sets of atomic propositions as before, *i.e.*, by taking $\mathcal{R} = \{(m, m') \mid m \xrightarrow{t} m'\}$ when considering invisibility of t. Finally, we introduce another way to lift invisibility to structural transitions: t is *strongly q-invisible* iff the set $\{(m, m') \mid \forall p \in P : m'(p) = m(p) + W(t, p) - W(p, t)\}$ is q-invisible. Strong invisibility does not take the presence of a transition $m \xrightarrow{t} m'$ into account, and purely reasons about the effects of t. Value invisibility and strong invisibility are new in the current work, although strong invisibility was inspired by the notion of invisibility that is proposed by Varpaaniemi in [22].

$$\begin{tikzcd}
 & \mathcal{I}_{sv} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathcal{I}_s \arrow[d] \arrow[dr] & \mathcal{I}^r_{sv} \arrow[dl] \arrow[dr] & \mathcal{I}_v \arrow[d] \arrow[dl] \\
\mathcal{I}^r_s \arrow[dr] & \mathcal{I} \arrow[d] & \mathcal{I}^r_v \arrow[dl] \\
 & \mathcal{I}^r &
\end{tikzcd}$$

Fig. 5: Lattice of sets of invisible actions. Arrows represent a subset relation.

We indicate the sets of all value, reach and strongly invisible structural transitions with $\mathcal{I}_v$, $\mathcal{I}^r$ and $\mathcal{I}_s$ respectively. Since $\mathcal{I}_v \subseteq \mathcal{I}$, $\mathcal{I}_s \subseteq \mathcal{I}$ and $\mathcal{I} \subseteq \mathcal{I}^r$, the set of all their possible combinations forms the lattice shown in Figure 5. In the remainder, the weak equivalence relations that follow from each of the eight invisibility notions are abbreviated, *e.g.*, $\sim_{\mathcal{I}^r_{sv}}$ becomes $\sim^r_{sv}$.



$$\begin{tikzcd}
 & \sim_{sv} \arrow[dl] \arrow[d] \arrow[dr] & & & \triangleq \arrow[d] \\
\sim_s \arrow[d] \arrow[dr] & \sim^r_{sv} \arrow[dl] \arrow[dr] & \sim_v \arrow[dl] \arrow[d] \arrow[rr, dotted, "\text{Theorem 5}"] & & \triangleq_p \arrow[d] \\
\sim^r_s \arrow[dr] \arrow[uurrrr, dotted, "\text{Theorem 6}"] & \sim \arrow[d] & \sim^r_v \arrow[dl] \arrow[rr, dotted, "\text{Theorem 4}"] & & \triangleq_l \\
 & \sim^r & & &
\end{tikzcd}$$

Fig. 6: Two lattices containing variations of weak equivalence and stutter equivalence, respectively. Solid arrows indicate a subset relation inside the lattice; dotted arrows follow from the indicated theorems and show when the LSTS of a Petri net is labelled consistently.

Example 3. Consider again the Petri net and LSTS from Example 2. We can define q_l and q_p as linear and polynomial propositions, respectively:

– $q_l := p_3 + p_4 + p_6 = 0$ is a linear proposition, which holds when neither p_3, p_4 nor p_6 contains a token. Structural transition t is q_l-invisible, because $m \xrightarrow{t} m'$ implies that $m(p_3) = m'(p_3) \geq 1$, and thus neither m nor m is labelled with q_l. On the other hand, t is not value q_l-invisible (by the transition $101100 \xrightarrow{t} 101010$) or strongly reach q_l-invisible (by 010100 and 010010). However, t_{key} is strongly value q_l-invisible: it moves a token from p_4 to p_6 and hence never changes the value of $p_3 + p_4 + p_6$.
– $q_p := (1 - p_3)(1 - p_5) = 1$ is a polynomial proposition, which holds in all reachable markings m where $m(p_3) = 0$ and $m(p_5) = 0$. Structural transition t is reach value q_p-invisible, but not q_p-invisible (by $002120 \xrightarrow{t} 002030$) or strongly reach q_p invisible. Strong value q_p-invisibility of t_{key} follows immediately from the fact that the adjacent places of t_{key}, *viz.* p_4 and p_6, do not occur in the definition of q_p.

This yields the state labelling which is shown in Example 2. □

Given a weak equivalence relation $R_\sim$ and a stutter equivalence relation $R_\triangleq$, we write $R_\sim \preceq R_\triangleq$ to indicate that $R_\sim$ and $R_\triangleq$ yield consistent labelling. We spend the rest of this section investigating under which notions of invisibility and propositions from the literature, the LSTS of a Petri net is labelled consistently. More formally, we check for each weak equivalence relation $R_\sim$ and each stutter equivalence relation $R_\triangleq$ whether $R_\sim \preceq R_\triangleq$. This tells us when existing stubborn set theory can be applied without problems. The two lattices containing all weak and stuttering equivalence relations are depicted in Figure 6; each dotted arrow represents a consistent labelling result. Before we continue, we first introduce an auxiliary lemma.

Lemma 5. *Let I be a set of invisible structural transitions and L some labelling function. If for all $t \in I$ and paths $\pi = m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots$ and $\pi' = m_0 \xrightarrow{t} m'_0 \xrightarrow{t_1} m'_1 \xrightarrow{t_2} \dots$, it holds that $\pi \triangleq_L \pi'$, then $\sim_I \preceq \triangleq_L$.*

Proof. We assume that the following holds for all paths and $t \in I$:

$$m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \cdots \triangleq_L m_0 \xrightarrow{t} m'_0 \xrightarrow{t_1} m'_1 \xrightarrow{t_2} \ldots \qquad (\dagger)$$

We consider two initial paths π and π' such that $\pi \sim_I \pi'$ and prove that $\pi \triangleq_L \pi'$. The proof proceeds by induction on the combined number of invisible structural transitions (taken from I) in π and π'. In the base case, π and π' contain only visible structural transitions, and $\pi \sim_I \pi$' implies $\pi = \pi'$ since Petri nets are deterministic. Hence, $\pi \triangleq_L \pi'$.

For the induction step, we take as hypothesis that, for all initial paths π and π' that together contain at most k invisible structural transitions, $\pi \sim_I \pi'$ implies $\pi \triangleq_L \pi'$. Let π and π' be two arbitrary initial paths such that $\pi \sim_I \pi'$ and the total number of invisible structural transitions contained in π and π' is k. We consider the case where an invisible structural transition is introduced in π', the other case is symmetric. Let $\pi' = \sigma_1\sigma_2$ for some σ_1 and σ_2. Let $t \in I$ be some invisible structural transition and $\pi'' = \sigma_1 t \sigma'_2$ such that σ_2 and σ'_2 contain the same sequence of structural transitions. Clearly, we have $\pi' \sim_I \pi''$. Here, we can apply our original assumption ($\dagger$), to conclude that $\sigma_2 \triangleq t\sigma'_2$, *i.e.*, the extra stuttering step t thus does not affect the labelling of the remainder of π''. Hence, we have $\pi' \triangleq_L \pi''$ and, with the induction hypothesis, $\pi \triangleq_L \pi''$. Note that π and π'' together contain $k+1$ invisible structural transitions.

In case π and π' together contain an infinite number of invisible structural transitions, $\pi \sim_I \pi'$ implies $\pi \triangleq_L \pi'$ follows from the fact that the same holds for all finite prefixes of π and π' that are related by $\sim_I$. □

The following theorems each focus on a class of atomic propositions and show which notion of invisibility is required for the LSTS of a Petri net to be labelled consistently. In the proofs, we use a function d_t, defined as $d_t(p) = W(t,p) - W(p,t)$ for all places p, which indicates how structural transition t changes the state. Furthermore, we also consider functions of type $P \to \mathbb{N}$ as vectors of type $\mathbb{N}^{|P|}$. This allows us to compute the pairwise addition of a marking m with d_t ($m + d_t$) and to indicate that t does not change the marking ($d_t = 0$).

Theorem 4. *Under reach value invisibility, the LSTS underlying a Petri net is labelled consistently for linear propositions,* i.e., $\sim^r_v \preceq \triangleq_l$.

Proof. Let $t \in \mathcal{I}^r_v$ be a reach value invisible structural transition such that there exist reachable markings m and m' with $m \xrightarrow{t} m'$. If such a t does not exist, then $\sim^r_v$ is the reflexive relation and $\sim^r_v \preceq \triangleq_l$ is trivially satisfied. Otherwise, let $q := f(p_1, \ldots, p_n) \bowtie k$ be a linear proposition. Since t is reach value invisible and f is linear, we have $f(m) = f(m') = f(m + d_t) = f(m) + f(d_t)$ and thus $f(d_t) = 0$. It follows that, given two paths $\pi = m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots$ and $\pi' = m_0 \xrightarrow{t} m'_0 \xrightarrow{t_1} m'_1 \xrightarrow{t_2} \ldots$, the addition of t does not influence f, since $f(m_i) = f(m_i) + f(d_t) = f(m_i + d_t) = f(m'_i)$ for all i. As a consequence, t also does not influence q. With Lemma 5, we deduce that $\sim^r_v \preceq \triangleq_l$. □

Whereas in the linear case one can easily conclude that π and π' are stutter equivalent under f, in the polynomial case, we need to show that f is constant under all value invisible structural transitions t, even in markings where t is not enabled. This follows from the following proposition.

Proposition 1. *Let $f : \mathbb{N}^n \to \mathbb{Z}$ be a polynomial function, $a, b \in \mathbb{N}^n$ two constant vectors and $c = a - b$ the difference between a and b. Assume that for all $x \in \mathbb{N}^n$ such that $x \geq b$, where $\geq$ denotes pointwise comparison, it holds that $f(x) = f(x + c)$. Then, f is constant in the vector c,* i.e., *$f(x) = f(x + c)$ for all $x \in \mathbb{N}^n$.*

Proof. Let f, a, b and c be as above and let $\mathbf{1} \in \mathbb{N}^n$ be the vector containing only ones. Given some arbitrary $x \in \mathbb{N}^n$, consider the function $g_x(t) = f(x + t \cdot \mathbf{1} + c) - f(x + t \cdot \mathbf{1})$. For sufficiently large t, it holds that $x + t \cdot \mathbf{1} \geq b$, and it follows that $g_x(t) = 0$ for all sufficiently large t. This can only be the case if g_x is the zero polynomial, *i.e.*, $g_x(t) = 0$ for all t. As a special case, we conclude that $g_x(0) = f(x + c) - f(x) = 0$. □

The intuition behind this is that $f(x + c) - f(x)$ behaves like the directional derivative of f with respect to c. If the derivative is equal to zero in infinitely many x, f must be constant in the direction of c. We will apply this result in the following theorem.

Theorem 5. *Under value invisibility, the LSTS underlying a Petri net is labelled consistently for polynomial propositions,* i.e., $\sim_v \preceq \triangleq_p$.

Proof. Let $t \in \mathcal{I}_v$ be a value invisible structural transition, m and m' two markings with $m \xrightarrow{t} m'$, and $q := f(p_1, \ldots, p_n) \bowtie k$ a polynomial proposition. Note that infinitely many such (not necessarily reachable) markings exist in $\mathcal{M}$, so we can apply Proposition 1 to obtain $f(m) = f(m + d_t)$ for all markings m. It follows that, given two paths $\pi = m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots$ and $\pi' = m_0 \xrightarrow{t} m_0' \xrightarrow{t_1} m_1' \xrightarrow{t_2} \ldots$, the addition of t does not alter the value of f, since $f(m_i) = f(m_i + d_t) = f(m_i')$ for all i. As a consequence, t also does not change the labelling of q. Application of Lemma 5 yields $\sim_v \preceq \triangleq_p$. □

Varpaaniemi shows that the LSTS of a Petri net is labelled consistently for arbitrary propositions under his notion of invisibility [22, Lemma 9]. Our notion of strong visibility, and especially strong reach invisibility, is weaker than Varpaaniemi's invisibility, so we generalise the result to $\sim_s^r \preceq \triangleq$.

Theorem 6. *Under strong reach visibility, the LSTS underlying a Petri net is labelled consistently for arbitrary propositions,* i.e., $\sim_s^r \preceq \triangleq$.

Proof. Let $t \in \mathcal{I}_s^r$ be a strongly reach invisible structural transition and $\pi = m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots$ and $\pi' = m_0 \xrightarrow{t} m_0' \xrightarrow{t_1} m_1' \xrightarrow{t_2} \ldots$ two paths. Since, $m_i' = m_i + d_t$ for all i, it holds that either (i) $d_t = 0$ and $m_i = m_i'$ for all i; or (ii) each pair (m_i, m_i') is contained in $\{(m, m') \mid \forall p \in P : m'(p) = m(p) + W(t, p) - W(p, t)\}$, which is the set that underlies strong reach invisibility of t. In both cases, $L(m_i) = L(m_i')$ for all i. It follows from Lemma 5 that $\sim_s^r \preceq \triangleq$. □

To show that the results of the above theorems cannot be strengthened, we provide two negative results.

Theorem 7. *Under ordinary invisibility, the LSTS underlying a Petri net is not necessarily labelled consistently for arbitrary propositions,* i.e., $\sim \not\subseteq \triangleq$.

Proof. Consider the Petri net from Example 2 with the arbitrary proposition q_l. Disregard q_p for the moment. Structural transition t is q_l-invisible, hence the paths corresponding to $t_1t_2tt_3$ and $tt_1t_2t_3$ are weakly equivalent under ordinary invisibility. However, they are not stutter equivalent. □

Theorem 8. *Under reach value invisibility, the LSTS underlying a Petri net is not necessarily labelled consistently for polynomial propositions,* i.e., $\sim_v^r \not\subseteq \triangleq_p$.

Proof. Consider the Petri net from Example 2 with the polynomial proposition $q_p := (1-p_3)(1-p_5) = 1$ from Example 3. Disregard q_l in this reasoning. Structural transition t is reach value q_p-invisible, hence the paths corresponding to $t_1t_2tt_3$ and $tt_1t_2t_3$ are weakly equivalent under reach value invisibility. However, they are not stutter equivalent for polynomial propositions. □

It follows from Theorems 7 and 8 and transitivity of $\subseteq$ that Theorems 4, 5 and 6 cannot be strengthened further. In terms of Figure 6, this means that the dotted arrows cannot be moved downward in the lattice of weak equivalences and cannot be moved upward in the lattice of stutter equivalences. The implications of these findings on related work will be discussed in the next section.

7 Related Work

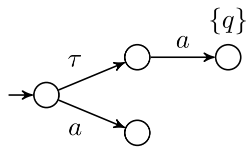
There are many works in literature that apply stubborn sets. We will consider several works that aim to preserve LTL$_{-X}$ and discuss whether they are correct when it comes to the problem presented in the current work.

Liebke and Wolf [10] present an approach for efficient CTL model checking on Petri nets. For some formulas, they can reduce CTL model checking to LTL model checking, which allows greater reductions under POR. They rely on the incorrect LTL preservation theorem, and since they apply the techniques on Petri nets with ordinary invisibility, their theory is incorrect (Theorem 7). Similarly, the overview of stubborn set theory presented by Valmari and Hansen in [21] applies reach invisibility and does not necessarily preserve LTL$_{-X}$. Varpaaniemi [22] also applies stubborn sets to Petri nets, but relies on a visibility notion that is stronger than strong invisibility. The correctness of these results is thus not affected (Theorem 6). The approach of Bønneland *et al.* [4] operates on two-player Petri nets, but only aims to preserve reachability and consequently does not suffer from the inconsistent labelling problem.

A generic implementation of weak stubborn sets is proposed by Laarman *et al.* [9]. They use abstract concepts such as guards and transition groups to implement POR in a way that is agnostic of the input language. The theory they present includes condition **D1**, which is too weak, but the accompanying implementation follows the framework of Section 4.1, and thus it is correct by Theorem 2 The implementations proposed in [21,23] are similar, albeit specific for Petri nets.

Others [6,8] perform action-based model checking and thus strive to preserve weak trace equivalence or inclusion. As such, they do not suffer from the problems discussed here, which applies only to state labels.

Although Beneš *et al.* [2,3] rely on ample sets, and not on stubborn sets, they also discuss weak trace equivalence and stutter-trace equivalence. In fact, they present an equivalence relation for traces that is a combination of weak and stutter equivalence. The paper includes a theorem that weak equivalence implies their new state/event equivalence [2, Theorem 6.5]. However, the counter-example on the right shows that this consistent labelling theorem does not hold. Here, the action τ is invisible, and the two paths in this transition system are thus weakly equivalent. However, they are not stutter equivalent, which is a special case of state/event equivalence. Although the main POR correctness result [2, Corollary 6.6] builds on the incorrect consistent labelling theorem, its correctness does not appear to be affected. An alternative proof can be constructed based on Lemmas 1 and 2.



The current work is not the first to point out mistakes in POR theory. In [14], Siegel presents a flaw in an algorithm that combines POR and on-the-fly model checking [12]. In that setting, POR is applied on the product of an LSTS and a Büchi automaton. Let q be a state of the LSTS and s a state of the Büchi automaton. While investigating a transition $(q, s) \xrightarrow{a} (q', s')$, condition **C3**, which—like condition **L**—aims to solve the action ignoring problem, incorrectly sets $r(q, s') = \mathit{enabled}(q)$ instead of $r(q, s) = \mathit{enabled}(q)$.

8 Conclusion

We discussed the inconsistent labelling problem for preservation of stutter-trace equivalence with stubborn sets. The issue is relatively easy to repair by strengthening condition **D1**. For Petri nets, altering the definition of invisibility can also resolve inconsistent labelling depending on the type of atomic propositions. The impact on applications presented in related works seems to be limited: the problem is typically mitigated in the implementation, since it is very hard to compute **D1** exactly. This is also a possible explanation for why the inconsistent labelling problem has not been noticed for so many years.

Since this is not the first error found in POR theory [14], a more rigorous approach to proving its correctness, *e.g.* using proof assistants, would provide more confidence.

References

1. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)

2. Beneš, N., Brim, L., Buhnova, B., Ern, I., Sochor, J., Vařeková, P.: Partial order reduction for state/event LTL with application to component-interaction automata. Science of Computer Programming **76**(10), 877–890 (2011). https://doi.org/10.1016/j.scico.2010.02.008
3. Beneš, N., Brim, L., Černá, I., Sochor, J., Vařeková, P., Zimmerova, B.: Partial Order Reduction for State/Event LTL. In: IFM 2009. LNCS, vol. 5423, pp. 307–321 (2009). https://doi.org/10.1007/978-3-642-00255-7_21
4. Bønneland, F.M., Jensen, P.G., Larsen, K.G., Muñiz, M.: Partial Order Reduction for Reachability Games. In: CONCUR 2019. vol. 140, pp. 23:1–23:15 (2019). https://doi.org/10.4230/LIPIcs.CONCUR.2019.23
5. Gerth, R., Kuiper, R., Peled, D., Penczek, W.: A Partial Order Approach to Branching Time Logic Model Checking. Information and Computation **150**(2), 132–152 (1999). https://doi.org/10.1006/inco.1998.2778
6. Gibson-Robinson, T., Hansen, H., Roscoe, A.W., Wang, X.: Practical Partial Order Reduction for CSP. In: NFM 2015. LNCS, vol. 9058, pp. 188–203 (2015). https://doi.org/10.1007/978-3-319-17524-9_14
7. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems, LNCS, vol. 1032. Springer (1996). https://doi.org/10.1007/3-540-60761-7
8. Hansen, H., Lin, S., Liu, Y., Nguyen, T.K., Sun, J.: Diamonds Are a Girl's Best Friend: Partial Order Reduction for Timed Automata with Abstractions. In: CAV 2014. LNCS, vol. 8559, pp. 391–406 (2014). https://doi.org/10.1007/978-3-319-08867-9_26
9. Laarman, A., Pater, E., van de Pol, J., Hansen, H.: Guard-based partial-order reduction. STTT **18**(4), 427–448 (2016). https://doi.org/10.1007/s10009-014-0363-9
10. Liebke, T., Wolf, K.: Taking Some Burden Off an Explicit CTL Model Checker. In: Petri Nets 2019. LNCS, vol. 11522, pp. 321–341 (2019). https://doi.org/10.1007/978-3-030-21571-2_18
11. Peled, D.: All from One, One for All: on Model Checking Using Representatives. In: CAV 1993. LNCS, vol. 697, pp. 409–423 (1993). https://doi.org/10.1007/3-540-56922-7_34
12. Peled, D.: Combining partial order reductions with on-the-fly model-checking. FMSD **8**(1), 39–64 (1996). https://doi.org/10.1007/BF00121262
13. Schmidt, K.: Stubborn sets for model checking the EF/AG fragment of CTL. Fundamenta Informaticae **43**(1-4), 331–341 (2000)
14. Siegel, S.F.: What's Wrong with On-the-Fly Partial Order Reduction. In: CAV 2019. LNCS, vol. 11562, pp. 478–495 (2019). https://doi.org/10.1007/978-3-030-25543-5_27
15. Valmari, A.: A Stubborn Attack on State Explosion. In: CAV 1990. LNCS, vol. 531, pp. 156–165 (1991). https://doi.org/10.1007/BFb0023729
16. Valmari, A.: Stubborn sets for reduced state space generation. In: Advances in Petri Nets. vol. 483, pp. 491–515 (1991). https://doi.org/10.1007/3-540-53863-1_36
17. Valmari, A.: A Stubborn Attack on State Explosion. Formal Methods in System Design **1**(4), 297–322 (1992). https://doi.org/10.1007/BF00709154
18. Valmari, A.: The state explosion problem. In: ACPN 1996. LNCS, vol. 1491, pp. 429–528 (1996). https://doi.org/10.1007/3-540-65306-6_21
19. Valmari, A.: Stubborn Set Methods for Process Algebras. In: POMIV 1996. DIMACS, vol. 29, pp. 213–231 (1997). https://doi.org/10.1090/dimacs/029/12
20. Valmari, A.: Stop It, and Be Stubborn! TECS **16**(2), 46:1–46:26 (2017). https://doi.org/10.1145/3012279

21. Valmari, A., Hansen, H.: Stubborn Set Intuition Explained. In: ToPNoC XII. LNCS, vol. 10470, pp. 140–165 (2017). https://doi.org/10.1007/978-3-662-55862-1_7
22. Varpaaniemi, K.: On Stubborn Sets in the Verification of Linear Time Temporal Properties. FMSD **26**(1), 45–67 (2005). https://doi.org/10.1007/s10703-005-4594-y
23. Wolf, K.: Petri Net Model Checking with LoLA 2. In: Petri Nets 2018. LNCS, vol. 10877, pp. 351–362 (2018). https://doi.org/10.1007/978-3-319-91268-4_18