Büchi-Automata guided
Partial Order Reduction for LTL

Torsten Liebke

Universität Rostock, Institut für Informatik, Germany
torsten.liebke@uni-rostock.de

Abstract. Partial order reduction (POR) is a key technique to tackle the state explosion problem in the model checking (MC) domain. POR is based on the observation that concurrent and independent running processes contribute extensively to the state explosion problem, while having only little influence on the property preservation of individual processes. In essence, while building the state space, in each found state, POR methods compute a subset of transitions and only fire the transitions in it, to explore more states. Hence, the state space is reduced. The computed subset of transition has to satisfy certain requirements for property preservation. We propose a new POR method for linear temporal logic (LTL), which generalizes and extends ideas from [8]. LTL MC uses most commonly Büchi automata to represent the property under investigation. Compared to conventional LTL POR methods we exploit additional information available from the Büchi automaton. As a result, we are able to weaken or completely drop certain requirements and restrictions. For example the restriction to LTL$_{-\mathbf{X}}$ is dropped. We demonstrate the reduction efficiency on several examples.

Keywords: LTL · Model Checking · Partial Order Reduction · Stubborn Sets.

1 Introduction

Model checking (MC) is concerned with the question, whether a given model of a system meets a given specification. Although there has been remarkable progress in verifying properties over the years, the biggest issue for MC remains the state explosion problem. In explicit MC one core technique to tackle large state spaces is the reduction of the original state space to a smaller one. Example methods are symmetry [14] or partial order reduction (POR) [3, 10, 17]. They both have the ability to preserve the specification under investigation while reducing the state space substantially.

POR appears to be the most powerful reduction technique. Our explicit model checker LoLA 2 [24], which is regularly a guest on the podium of the yearly model checking contest (MCC) [1], solves around 70 % of the queries in the *reachability* category without and around 90 % with POR techniques. Thus, from the remaining 30 % of the really hard-to-solve queries, two-thirds still can be solved using POR methods. The numbers for the linear temporal logic (LTL) category are similar.

POR is based on the observation that concurrent and independent running processes contribute extensively to the state explosion problem, while they have little influence on the property preservation of individual processes. There exist several approaches to POR techniques: Ample Sets [10], Persistent Sets [3] and Stubborn Sets [16–18]. In essence, while building the state space, in each found state, they all compute a subset of transitions, called *aps set*, and only fire the transitions in it to explore more states. Hence, the state space is reduced. There exist also different approaches to preserve entire classes of properties, e.g., Computational Tree Logic [2], LTL [18], or only certain properties, e.g., deadlocks [15], reachability [7], liveness and other standard properties [13], or frequently occurring CTL formulas [9]. For this, a set of requirements, which the *aps sets* have to satisfy, was introduced in the literature. Each requirement has a specific purpose for proving property preservation.

In conventional LTL MC with POR the state space is first reduced, and then, together with the Büchi automaton of the (negated) formula, a product automaton is built, where the actual verification takes place. In [8] it was proposed, to first build the product automaton with the original state space, and then reduce the product automaton with the additional information available from the Büchi automaton. To the best of our knowledge, the idea, to use information from the Büchi automaton to reduce the state space, was first introduced in [11]. In general, the Büchi automata have very few states compared to the number of states of the system. The formulas in the MCC, although artificial, have rarely more than 5 – 10 Büchi states. Which is also consistent with our experience of real world use cases using our model checker LoLA 2.

With the additional information available from the Büchi automaton, we propose a new automata based POR, which is a generalization and extension of the ideas presented in [6] and [8]. The main idea is to focus the reduction of the product automata to the current Büchi state, i.e., all considerations regarding the formula are done locally around the current Büchi state. This gives rise to the main principle we use to achieve additional reduction power: all transition sequences, which do not use transitions from the aps set, cannot leave the current Büchi state. Staying as long as possible in the same Büchi state has another advantage, namely we only care to satisfy the part of the formula that leaves us in the same Büchi state, and we do not care about the satisfaction of the rest of the formula. Using the main principle, and the restricted scope of the formula, we are able to weaken or drop several requirements involved in conventional POR. We show the effectiveness of the newly introduced automata based POR with several examples. We analyze the reduction power, and compare it with the one from the conventional POR.

The paper is organized as follows. We start in Section 2 with a brief introduction of the terminology of P/T nets and LTL. Then we introduce LTL MC using Büchi automata in Section 3. Section 4 is concerned with the POR, more precisely the stubborn set method and the principles involved in the preservation of certain properties. In Section 5 we introduce our new automata based POR approach. We continue in Section 6 with a comparison between the conventional method and the new one. We conclude with some remarks regarding related work and some thoughts on future work in Section 7.

2 Terminology

Definition 1 (Place/Transition Net). *A* place/transition net*, P/T net for short, consists of a finite set of* places P*, a finite set of* transitions T *where* $P \cap T = \emptyset$*, a set of* arcs $F \subseteq (P \times T) \cup (T \times P)$*, a* weight function $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ *where* $W(x, y) = 0$ *if and only if* $(x, y) \notin F$*, and an* initial marking $m_0 : P \to \mathbb{N}$*, where* marking *is a mapping* $m : P \longrightarrow \mathbb{N}$.

The behaviour of a P/T net is defined by the *transition rule*.

Definition 2 (Transition rule of a P/T net). *Let* $N = [P, T, F, W, m_0]$ *be a P/T net. Transition* $t \in T$ *is* enabled *in marking* m *if, for all* $p \in P$, $W(p, t) \leq m(p)$*. If* t *is enabled in* m*,* t *can* fire*, producing a new marking* m' *where, for all* $p \in P$, $m'(p) = m(p) - W(p, t) + W(t, p)$*. This firing relation is denoted as* $m \xrightarrow{t} m'$*. It can be extended to a* marking sequence*, also called* transition sequence*, by the following inductive scheme:* $m \xrightarrow{\varepsilon} m$ *(for the empty sequence* ε*), and* $m \xrightarrow{\omega} m' \wedge m' \xrightarrow{t} m'' \implies m \xrightarrow{\omega t} m''$ *(for a sequence* $\omega \in T^*$ *and a transition* $t \in T$*).*

Using the transition rule, a P/T net induces the *reachability graph*, also called the *state space* of the P/T net.

Definition 3 (Reachability graph of a P/T net). *The* reachability graph (M, E) *of a P/T net* N *has a set of vertices* M *that comprises all markings that are reachable by any sequence from the initial marking of* N*. Every element* $m \xrightarrow{t} m'$ *of the firing relation (*$t \in T$*) defines an edge* E *from* m *to* m' *annotated with* t*.*

Throughout the paper we are concerned with paths in the reachability graph.

Definition 4 (Path, Suffix). *Let* (M, E) *be the reachability graph of a P/T net. A* finite path *starting in state* m_1 *is a sequence* $m_1 \ldots m_n$ *of states where, for all* $i < n$, $(m_i, m_{i+1}) \in E$*. An* infinite path *starting in* m_1 *is an infinite sequence* $m_1 m_2 \ldots$ *where, for all* i, $(m_i, m_{i+1}) \in E$*. A path is a finite or infinite path. A path is* maximal *if it is infinite, or is a finite path* $m_1 \ldots m_n$ *where* m_n *is a* deadlock*, i.e. a state where, for all* $m \in M$, $(m_n, m) \notin E$*.*

For self-containedness we continue with the introduction of the syntax and semantics of the temporal logic LTL, although we are using it mainly for the experimental validation. All results in this paper rely on Büchi automata as such.

Definition 5 (Syntax of LTL). `TRUE`, `FALSE`, `DEADLOCK`, `FIREABLE(`t`)` *(for* $t \in T$*), and* $k_1 p_1 + \cdots + k_n p_n \leq k$ *(*$k_i, k \in \mathbb{Z}, p_i \in P$*) are atomic propositions* AP*. For a given set of* AP*, the temporal logic LTL is inductively defined as follows:*

- *Every* $\varphi \in AP$ *is a state formula;*
- *If* φ *and* ψ *are state formulas, so are* $(\varphi \wedge \psi)$*,* $(\varphi \vee \psi)$*, and* $\neg\varphi$*;*
- *Every state formula is a path formula;*
- *If* φ *and* ψ *are path formulas, so are* $(\varphi \wedge \psi)$*,* $(\varphi \vee \psi)$*,* $\neg\varphi$*,* $\mathbf{X}\,\varphi$*,* $\mathbf{F}\,\varphi$*,* $\mathbf{G}\,\varphi$*,* $(\varphi\,\mathbf{U}\,\psi)$*, and* $(\varphi\,\mathbf{R}\,\psi)$*;*

Definition 6 (Semantics of LTL). *A P/T net N satisfies an LTL formula φ if and only if every path $\omega = m\, m_1 m_2 \dots$ in the reachability graph of N, satisfies φ according to the following inductive scheme:*

- $\omega \models \texttt{TRUE}$, $\omega \not\models \texttt{FALSE}$*;*
- $\omega \models \texttt{FIREABLE}(t)$ *if t is enabled in m;*
- $\omega \models \texttt{DEADLOCK}$ *if there is no enabled transition in m;*
- $\omega \models k_1 p_1 + \dots + k_n p_n \leq k$ *if* $k_1 m(p_1) + \dots + k_n m(p_n) \leq k$*;*
- $\omega \models \neg\varphi$ *if* $m \not\models \varphi$*;*
- $\omega \models (\varphi \wedge \psi)$ *if* $m \models \varphi$ *and* $m \models \psi$*;*
- $\omega \models \mathbf{X}\varphi$ *if* $m_1 \models \varphi$*;*
- $\omega \models \varphi\mathbf{U}\psi$ *if there is exists a $k \geq 0$ s.t. that $m_k \models \psi$ and, for all i with $0 \leq i < k$, $m_i \models \varphi$.*

The semantics of the remaining LTL operators is defined using the tautologies $(\varphi \vee \psi) \iff \neg(\neg\varphi \wedge \neg\psi)$, $\mathbf{F}\varphi \iff (\texttt{TRUE}\ \mathbf{U}\varphi)$, $\mathbf{G}\varphi \iff \neg F \neg\varphi$, and $(\varphi\mathbf{R}\psi) \iff \neg(\neg\varphi\mathbf{U}\neg\psi)$.

LTL formulas can be stuttering invariant.

Definition 7 (Stuttering invariance). *An LTL formula φ is* stuttering invariant*, if for all finite paths ω_1, all markings m and all infinite paths ω_2 it holds: $\omega_1 m \omega_2 \models \varphi \iff \omega_1 m m \omega_2 \models \varphi$.*

Stuttering invariant LTL formulas are exactly those formulas that do not contain the $\mathbf{X}$-operator [12]. Note, the absence of the $\mathbf{X}$-operator is a sufficient but not a necessary condition for stuttering invariance. In the following we call the set of LTL formulas without the $\mathbf{X}$-operator LTL$_{-\mathbf{X}}$.

3 LTL Model Checking with Büchi Automata

For self-containedness we also briefly introduce the standard definition of Büchi automata, how they are used in MC, and their link to LTL. In [22] it is shown that every LTL formula φ, with size $|\varphi|$, can be translated to a Büchi automaton B_φ, with size $|B_\varphi| \leq 2^{|\varphi|}$, which accepts exactly those infinite runs that satisfy φ.

Definition 8 (Büchi automaton). *A* Büchi automaton $B = (Q, q_0, \delta, \lambda, Q_F)$ *consists of*

- *a finite set Q of Büchi states*
- *with $q_0 \in Q$ as its initial state,*
- *a transition relation $\delta \subseteq Q \times Q$*
- *with labelling function $\lambda : \delta \to \Phi$, where Φ are propositional logic formulas over a set of atomic propositions,*
- *and a set $Q_F \subseteq Q$ of final states.*

B accepts an infinite sequence of markings $m_1 m_2 \dots$ if and only if there is an infinite sequence $q_0 q_1 \dots$ s.t.

- *for all* i, $(q_i, q_{i+1}) \in \delta$;
- *for all* i, $m_{i+1} \models \lambda(q_i, q_{i+1})$;
- *for infinitely many* i, $q_i \in Q_F$.

The actual on-the-fly verification runs in the product automaton B^*, which is again a Büchi automaton. It is built from the system (M, E) and the Büchi automaton of the negated formula $B_{\neg\varphi}$.

Definition 9 (Product system). *Let* $B = (Q, q_s, \delta, \lambda, Q_F)$ *be a Büchi automata and* N *be a P/T net with reachability graph* (M, E)*. The* product system $(M, E) \cap B$ *is a Büchi automaton* $B^* = (Q^*, q_0^*, \delta^*, \lambda^*, Q_F^*)$ *where*

- q_0^* *is some element not contained in* $M \times Q$,
- $Q^* = M \times Q \cup \{q_0^*\}$,
- $(q_0^*, (m, q)) \in \delta^*$ *iff* $(q_0, q) \in \delta$ *and* $m = m_0 \models \lambda(q_0, q)$,
- $((m, q), (m', q')) \in \delta^*$ *iff* $(m, m') \in E$, $(q, q') \in \delta$ *and* $m' \models \lambda(q, q')$,
- $\lambda(q^*, q^{*\prime}) = tt$ *for all* $q^*, q^{*\prime} \in \delta^*$*, where* tt *is the formula which is always true,*
- $(m, q) \in Q_F^*$ *iff* $q \in Q_F$.

The product automaton represents a combination of the current marking in the reachability graph and the current state of the Büchi automaton. It accepts exactly those infinite paths which violate the property φ. This means that to verify an LTL formula φ, we check whether there exists a path in B^*, which can be executed in (M, E). If we found such a path, also called counterexample, it follows that φ does not hold in the system.

Proposition 1 (LTL to Büchi automaton, [22]). *For every* LTL *formula* φ*, there exists a Büchi automaton that accepts exactly those paths which violate* φ.

And the property φ is satisfied in the system, if the product automaton B^* accepts the empty language.

Proposition 2 (Emptiness check, [22]). *There exists an infinite path realizable in the reachability graph* (M, E) *of the P/T net* N *and accepted by* B *if and only if the product system* $(M, E) \cap B$ *has an accepting run.*

Note that also a reduced reachability graph can be used to construct the product automaton. I.e., MC with Büchi automata can be combined with other reduction techniques, e.g., POR.

4 Partial Order Reduction — the Stubborn Set Method

Partial order reduction (POR) is a technique to reduce the state space of a system, while preserving certain properties. There exist different approaches to this topic: Ample Sets [10], Persistent Sets [3], Stubborn Sets [16–18] and some more. In essence, while building the state space, in each found state, they all compute a *subset of transitions* and only fire the transitions in it, to explore more states. In the sequel we are concerned with the stubborn set approach. The reason for this is that stubborn sets push the theoretical limits to reach "as good reduction as possible, while ample and persistent sets have favoured straightforward easily implementable conditions and algorithms" [21]. Still, stubborn sets are easy enough to implement (for details see [23]) and two of the most successful tools in the yearly MCC, TAPAAL [4] and our tool LoLA 2 [24], are using them. This shows that computing stubborn sets in practice is fast.

Fig. 1 shows a simplified example, where we want to check whether m_3 satisfies some property φ. There are two paths, t_1t_2 and t_2t_1, and both reach the same marking m_3 independent of intermediate markings. This means in m_0 we can reduce the set of transitions, to explore more states, to either t_1 or t_2. Here we choose t_1. Since we do not follow t_2, we only reach the intermediate marking m_1, and from here with t_2 we reach m_3. The state space is reduced since we never explored m_2.

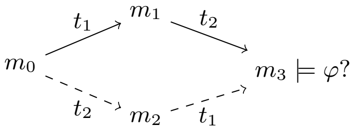


Fig. 1. Simplified reachability graph to check a property φ in m_3. Only the solid path needs to be explored with POR, and the dashed path can be omitted. Since t_2 is not fired in m_0, the marking m_1 is never used, and thus the state space is reduced.

Given is an arbitrary, fixed P/T net $N = [P, T, F, W, m_0]$ with reachability graph (M, E) and a property φ. The stubborn set method aims at producing a subgraph (M', E') of the original reachability graph (M, E), s.t. the evaluation of φ using (M', E') yields the same truth value as the evaluation of (M, E). We first define a mechanism to restrict the set of transitions T in each marking m to a subset called *stubborn set* or *stubborn(m)*.

Definition 10 (Stubborn set generator). *Let $N = [P, T, F, W, m_0]$ be a P/T net with reachability graph (M, E). A function* $\mathrm{stubborn} : M \to 2^T$ *is called* stubborn set generator *and produces the reduced reachability graph (M', E'), s.t. $(m, m') \in E' \iff m \xrightarrow{t} m'$ for a transition $t \in \mathrm{stubborn}(m)$. M' is the set of markings, which can be reached from the initial marking by only firing transitions from* $\mathrm{stubborn}(m)$ *in the marking m.*

The way the stubborn set generator is defined, gives us the possibility to choose in any marking any subset we want, there are no restrictions. This is certainly not useful if a property is investigated, since a random choice would most certainly not preserve the property. The goal would be to reduce the state space as much as possible, while choosing the subset in a way that the property under investigation is preserved. For this, a set of requirements, which we also call principles, were introduced in the literature. Each requirement has a specific purpose for proving property preservation. In most cases, we assume that there is a path π in the full reachability graph (e.g., a witness path or a counterexample for the property under investigation) and show that the reduced system contains a path π' that is equally fit w.r.t. the studied property. We first introduce the requirements, using the same terminology as in [9], before we continue to show which classes of properties are preserved.

The first principle is *commutativity* (COM for short). It is the heart of virtually all POR methods. It describes a specific independence between transitions in the stubborn set and transitions not in the stubborn set. The main purpose of COM is that π' may execute transitions in another order than π. The firing of transitions, which are not part of the stubborn set, can be deferred until later, without losing possible ongoing paths. The left side of Fig. 2 shows this situation.

Definition 11 (COM: The commutativity principle). $\mathrm{stubborn}(m) \subseteq T$ *satisfies the commutativity principle if, for all* $\omega \in (T\setminus \mathrm{stubborn}(m))^*$ *and all* $t \in \mathrm{stubborn}(m)$, $m \xrightarrow{\omega t} m'$ *implies* $m \xrightarrow{t\omega} m'$.

$$\begin{tikzcd}
m \arrow[r, "\omega"] \arrow[d, "t"'] & m'' \arrow[d, "t"'] \\
m''' \arrow[r, "\omega"] & m'
\end{tikzcd}
\qquad
\begin{tikzcd}
m \arrow[r, "\omega"] \arrow[d, "t"'] & m' \arrow[d, "t"'] \\
{} & {}
\end{tikzcd}$$

Fig. 2. Graphical representation of COM (left) and KEY (right).

The next requirement is the *key transition principle* (KEY for short). KEY ensures that there is an enabled transition which can be switched to the beginning of a transition sequence. The purpose of KEY (in connection with COM) is that π' may contain transitions that are not occurring in π. In more detail KEY states that there is a transition t in the stubborn set which is enabled before and after firing a transition sequence which is not part of the stubborn set. This is illustrated in the right side of Fig. 2,

Definition 12 (KEY: The key transition principle). $\mathrm{stubborn}(m) \subseteq T$ *satisfies the key transition principle if in* m *no transition is enabled or it contains a transition* t^* *(a* key transition*) such that, for all* $\omega \in (T \setminus \mathrm{stubborn}(m))^*$, $m \xrightarrow{\omega} m'$ *implies that* t^* *is enabled in* m'.

Before we continue with the next requirement which is concerned with the property φ under investigation, we introduce the *invisibility property*. Transitions that can change the truth value of an atomic proposition ϕ occurring in φ are called visible. Transition which can not change any ϕ are called invisible. This means firing a sequence of invisible transitions will have no effect on the truth value of the property φ.

Definition 13 (Invisibility property). *A transition* t *is invisible w.r.t. an LTL formula* φ *regarding a set of transitions* $T' \subseteq T$, *if for all atomic propositions* ϕ *occurring in* φ, *all markings* $m \in M$ *and all finite transition sequences* $\omega \in (T')^*$ *it holds:*

$(m \xrightarrow{\omega} m' \wedge m \xrightarrow{t\omega} m'') \implies (m' \models \phi \iff m'' \models \phi)$. Otherwise the transition t is called visible w.r.t. φ regarding T'.

Now we can introduce the *visibility principle* (VIS(φ) for short). With VIS, visible transitions in π' appear in the same order as in π, if they appear in π'.

Definition 14 (VIS: The visibility principle). $\mathrm{stubborn}(m) \subseteq T$ *satisfies the visibility principle for a property φ if* $\mathrm{stubborn}(m)$ *contains only invisible transitions w.r.t. φ regarding* $(T \setminus \mathrm{stubborn}(m))$*, or all transitions.*

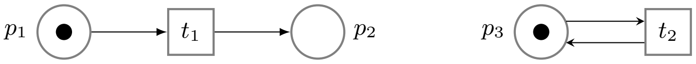


Fig. 3. Given is the formula $\varphi = \mathbf{F}(p_2 > 0)$. In the P/T net we choose $\{t_2\}$ as stubborn set in m_0. Since t_2 leads back to m_0 we chose over and over again the same stubborn set, $\{t_2\}$. This results in infinite stuttering, while we ignore t_1 indefinitely.

VIS(φ) ensures the same ordering of transitions, but it can introduce stuttering. As an example assume that the property $\varphi = \mathbf{F}(p_2 > 0)$ should hold in the P/T net from Fig. 3. Transition t_1 can change the truth value of φ, namely if t_1 fires it produces one token on p_2, and with this φ is satisfied. This means there are two valid stubborn sets in m_0. The first one is $\{t_2\}$ which contains only invisible transitions and the second one is $\{t_1, t_2\}$. If we choose $\{t_2\}$ as stubborn set, we reach the same marking m_0 again after firing t_2. We could now choose infinitely often t_2 as stubborn set, which results in stuttering. To avoid infinite stuttering we introduce the last requirement, the *non-ignoring principle* (IGN for short). IGN is used to ensure that all transitions of π are eventually occurring in π'. The concept behind this is that if marking m is the start of a cycle in the reachability graph, which results again in m, then in at least one marking m' in the cycle all enabled transitions have to be explored. In other words $\mathrm{stubborn}(m)$ consists of all enabled transitions in m' to possibly leave the cycle.

Definition 15 (IGN: The non-ignoring principle). $\mathrm{stubborn}$ *satisfies the non-ignoring principle if every cycle in the reduced reachability graph contains a marking where all enabled transitions are explored.*

There exist some more principles, but they are not needed for this paper. More details regarding principles and property preservation are provided in [7, 19, 21]. Combining these principle in a certain way results in property preservation of certain classes. The first property we look into is *deadlock*. Here no formula is needed and thus the visibility principle is also not needed. Furthermore, no cycle detection is required. This leaves us with COM and KEY.

Proposition 3 (Preservation of deadlocks, [17]). *If the principles COM and KEY are satisfied then (M', E') contains all deadlocks and at least one infinite path of the original reachability graph.*

For the preservation of terminal strongly connected components (SCC), used for example in the verification of liveness properties, we need to avoid infinite stuttering and thus we need IGN.

Proposition 4 (Preservation of terminal SCC, [20]). *If the principles COM, KEY, and IGN are satisfied then (M', E') contains at least one marking of every terminal SCC of the original reachability graph.*

With these two propositions we have almost all ingredients for the preservation of linear time properties, with which we are concerned in this paper. Next to COM, KEY, and IGN we also need to preserve the linear time property φ. Furthermore, we have to restrict LTL to stuttering invariant formulas. We call the following proposition the *conventional LTL$_{-\mathbf{X}}$* POR method.

Proposition 5 (Preservation of LTL$_{-\mathbf{X}}$, [10, 19]). *Let φ be an LTL property not using the $\mathbf{X}$-operator. If the principles COM, KEY, VIS(φ), and IGN are satisfied then φ is preserved.*

As mentioned before, the stubborn set generator used in conventional LTL$_{-\mathbf{X}}$ is a function $\mathrm{stubborn} : M \to 2^T$. This means the state space of the P/T net is first reduced and then the product automaton is built with the reduced state space. Compared to this, the authors in [8] propose to use also the Büchi automaton of the formula for the stubborn set generator. Therefore, the function changes to $\mathrm{stubborn} : M \times Q \to 2^T$. The Büchi automaton contains extra information, which can be used to weaken some stubborn set requirements. With weaker requirements we have better options to choose the transition set and thus, the chance of better reduction. The concept is to build on-the-fly the product automaton with the original reachability graph and then reduce the product automaton using the additional information from the Büchi automaton. POR introduced in [8] was only useable on a small class of Büchi automata, namely elementary Büchi automata. Elementary Büchi automata consist of a non-branching sequence of states, which may or may not have self loops, and the last of which is a final state. Although there are interesting properties in this class, like $\mathbf{G}(\varphi \implies \mathbf{F}\psi)$, it is still a rather large restriction.

5 Automata based Partial Order Reduction for LTL

In this section we introduce our new Büchi automata based POR. It is a generalization and extension of the ideas proposed in [6] and [8]. The section is inspired by [6]. As starting point, we use the stubborn set generator from [8], $\mathrm{stubborn} : M \times Q \to 2^T$, and the conventional LTL POR (proposition 5), which uses the principles COM, KEY, VIS(φ), and IGN to preserve LTL$_{-\mathbf{X}}$. The goal is to weaken or drop these principles to improve the reduction efficiency. Let us start with the main idea: We restrict the scope of the formula under investigation to the current Büchi state.

The representation of the formula is realized by the state transitions between the Büchi states. All considerations regarding the formula are done locally around the current Büchi state. To illustrate this, let us consider the example Büchi automaton from

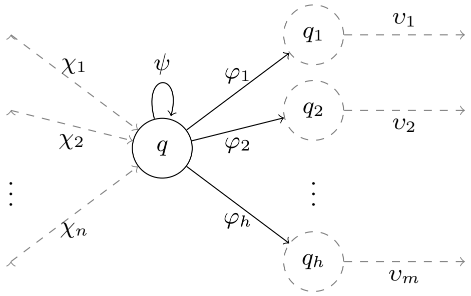


Fig. 4. The scope of the formula is restricted to the current Büchi state q, namely its retarding formula ψ, and its progressing formulas $\varphi_i, i \in [1, h]$.

Fig. 4. If we are in state q, we restrict the scope of the formula to the self loop ψ, and to the progressing formulas $\varphi_i, i \in [1, h]$. When we are in q, we do not care about all the other parts of the formula, i.e., $\chi_j, j \in [1, n]$, and $\upsilon_k, k \in [1, m]$. This brings us to the main principle for the reduction. We call it the *not-leaving principle* (NLG for short): *all transition sequences, which do not use transitions from the stubborn set, cannot leave the current Büchi state.*

To formalize this, let in the sequel $N = [P, T, F, W, m_0]$ be an arbitrary, fixed P/T net with reachability graph (M, E), and φ be an arbitrary LTL formula with its corresponding Büchi automaton $B_{\neg\varphi} = (Q, q_0, \delta, \lambda, Q_F)$. The formula φ can consists of several atomic sub-propositions ϕ. Let $q \in Q$ be a Büchi state and $\psi = \lambda(q, q)$ be the formula of the self loop $q \rightarrow q$. We call the self loop the *retarding formula*. Further let h be the number of outgoing arcs from q which are progressing to new Büchi states and let φ_i, with $0 \leq i \leq h$ be the formulas of the outgoing arcs. We call such formulas the *progressing formulas*. Now we formalize the main principle for the reduction.

Definition 16 (NLG: The not-leaving principle). $\mathrm{stubborn}(m, q) \subseteq T$ *satisfies the not-leaving principle if, for all* $i \in [1, h]$ *and for all* $\omega \in (T \setminus \mathrm{stubborn}(m, q))^+$, $m \xrightarrow{\omega} m'$ *implies* $m' \not\models \varphi_i$.

It holds that no transition sequence, from the set of transitions which are not in the stubborn set, can satisfy a progressing formula. For example, assume the progressing formula is $\varphi_i = p < 1$, which means that place p has less than one token on it. Then no transition that has p in its postset can be outside of the stubborn set, since it would change φ to true. With NLG some requirements from the conventional LTL$_{-\mathbf{X}}$ method can be weakened or even dropped, resulting in several advantages of the new stubborn set method.

Invisibility principle: Since all considerations regarding the formula are done locally around the current Büchi state, all other parts of the formula can be ignored. In fact, the invisibility property can be simplified. In the conventional method the influence on atomic sub-propositions regarding the result of the entire formula is dependent on the temporal progression and not further known. This means, the truth values of atomic sub-propositions must be preserved in both directions, $(m \models \psi \iff m' \models \psi)$.

Whereas in the automata based reduction all temporal operators are expressed in the state transitions of the Büchi automaton and with this, the influence of the atomic sub-proposition is determined. A satisfied formula allows a progression to another state q', but has no influence on other state transitions. It follows that we can simplify the invisibility property to an implication, $(m \models \psi \implies m' \models \psi)$. This means that a transition, which changes ψ from true to false, is not allowed, but a transition, which makes ψ only truer, is allowed now.

Definition 17 (Semi-invisibility property). *A transition $t \in T$ is called* semi-invisible *w.r.t. an LTL formula φ regarding a set of transitions $T' \subseteq T$, if for all markings $m \in M$, all finite transition sequences $\omega \in (T')^*$ and all atomic sub-propositions ϕ of φ it holds: $(m \xrightarrow{\omega} m' \wedge m \xrightarrow{t\omega} m'') \implies (m' \models \phi \implies m'' \models \phi)$. Otherwise the transition t is semi-visible with respect to ϕ regarding T'.*

As a consequence we can introduce also a weaker visibility principle, called the semi-invisibility principle (S-INV for short).

Definition 18 (S-INV: Semi-invisibility principle). $\mathrm{stubborn}(m, q) \subseteq T$ *satisfies the* semi-invisibility principle *for an LTL formula φ, if all enabled transitions $t \in \mathrm{stubborn}(m, q)$ are semi-invisible with respect to φ regarding $(T \backslash \mathrm{stubborn}(m, q))$, or all transitions $t \in (T \setminus \mathrm{stubborn}(m, q))$ are semi-invisible with respect to φ regarding the empty set.*

Non-ignoring principle and LTL$_{-\mathbf{X}}$: Since the reduction is only applied within a Büchi state, stuttering becomes irrelevant. Finite stuttering within a Büchi state does not change the accepting behaviour, and infinite stuttering can always be avoided, if this was possible in the original product automaton, due to the reduction principle. As consequence, the non-ignoring principle can be dropped. Furthermore, due to the irrelevance of stuttering, the restriction to LTL$_{-\mathbf{X}}$ can also be dropped. The introduced method preserves the full class of LTL. Due to the additional information available from the Büchi automaton and due to the fact, that the reduction is only applied to the current Büchi state, we know, whether we are in a transient Büchi state or not. Transient Büchi states can only happen in the initial state, when an atomic proposition is checked without any temporal operators, or they appear as representation of $\mathbf{X}$-operators of the formula. In [8] the additional information from the Büchi automaton was also used, but the reduction principle was different. It was not reduced to the current Büchi state and therefore they could not drop the $\mathbf{X}$-operator.

Key-transition principle: In accepting states infinite stuttering is possible and also desired. But we have to ensure that there always exists an enabled transition to prolong the path to an infinite path. Hence, KEY has to hold only in accepting states. If the current Büchi state is not accepting, then only transition sequences from the stubborn set can leave the current Büchi state. Since we are not in an accepting state, this implies that on the considered path we will change at some point in the future the Büchi state. It follows that in non-accepting states KEY does not has to hold, due to the fact that every transition sequence, which leaves the current Büchi state, always contains a transition from the stubborn set. This follows directly from NLG.

The drawback for the weakened or dropped requirements is that we have to uphold the main principle, that all transition sequences, which do not use transitions from the stubborn set, cannot leave the current Büchi state. Since the Büchi automaton for the LTL formula, compared to the transition system, is very small, we want to stay as long as possible in one Büchi state to achieve further reduction. This means that we want to avoid transitions which are progressing to the next Büchi state. As long as the progressing formulas φ_i are not satisfied, we are staying in the same Büchi state and can reduce further. We are now ready to state the main result of this paper.

Theorem 1 (Büchi automata based partial order reduction). *Given is a Büchi automaton $B = (Q, q_s, \delta, \lambda, Q_F)$ and a P/T net $N = (P, T, F, W, m_s)$ with reachable markings M. Further let* $\text{stubborn} : M \times Q \to 2^T$ *be a stubborn set generator which satisfies the following properties:*

1. $\text{stubborn}(m, q)$ *satisfies COM*
2. $\text{stubborn}(m, q)$ *satisfies S-INV with respect to* ψ
3. $\text{stubborn}(m, q) = T$ *or* $\forall t \in \text{stubborn}(m, q) : m \xrightarrow{t} m^{'} \implies m^{'} \models \psi$
4. $\text{stubborn}(m, q)$ *satisfies NLG*
5. *If* $q \in Q_F$*, then* $\text{stubborn}(m, q)$ *satisfies KEY*

Here $\psi = \lambda(q, q)$ *means the formula of the self loop* $q \to q$*, which is also called the retarding formula, and* $\forall i \in [1, h] : \varphi_i = \lambda(q, q_i), q \neq q_i$ *are the progressing formulas of the outgoing edges of* q *in* B*, where* h *is the actual number of outgoing edges.*

There exists an infinite accepting path in the reduced product system $\underline{\mathcal{P}}$*, which is generated by* stubborn*, if and only if there exists an infinite accepting path in the original product system* $\mathcal{P}$*.*

Proof. $\Longrightarrow$: The reduced state space $\underline{\mathcal{P}}$ is a subsystem of the original state space $\mathcal{P}$, this means accepting paths in $\underline{\mathcal{P}}$ trivially imply the same accepting paths in $\mathcal{P}$.

$\Longleftarrow$: Let $\pi \in (M \times Q)^\infty$ be an infinite accepting path in $\mathcal{P}$. Assume $\underline{\mathcal{P}}$ has no infinite accepting path. We can separate π in $\pi_1\pi_2$, s.t. π_1 is the largest prefix, which can be executed in $\underline{\mathcal{P}}$. Let (m, q) be the last state in π_1. Furthermore, let $(m_i, q_i), i \in \mathbb{N}_0$ be the state sequence in π_2 and $t_0t_1\dots$ the sequence of transitions, which induce π_2 from (m, q), s.t. $m \xrightarrow{t_0\dots t_i} m_i$.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q)}_{\pi_1}\underbrace{\xrightarrow{t_0} (m_0, q_0) \xrightarrow{t_1\dots t_i} (m_i, q_i)}_{\pi_2}$$

Since (m, q) is the last state in π_1, it follows that $t_0 \notin \text{stubborn}(m, q)$. Due to $t_0 \notin \text{stubborn}(m, q)$, requirement 4 (NLG), implies that $q = q_0$. We consider two cases. In the first one q lies in the accepting set of B and we do not change the Büchi state any more and in the second one the Büchi state is changed at least once more on the remaining path.

Case 1: *$\forall i \in \mathbb{N}_0 : q_i = q$:*

All states in π_2 remain in the same Büchi state q, i.e., $\forall i \in \mathbb{N}_0 : q_i = q$. This means that q is in the accepting set of B and all markings in π_2 satisfying the retarding formula

$\forall i \in \mathbb{N}_0 : m_i \models \psi$.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q)}_{\pi_1} \underbrace{\xrightarrow{t_0} (m_0, q) \xrightarrow{t_1 \ldots t_i} (m_i, q)}_{\pi_2}$$

We consider two sub-cases. In the first one we assume that there exists a transition t_i in $\mathrm{stubborn}(m, q)$ and in the second one we assume that there is no such t_i in $\mathrm{stubborn}(m, q)$.

Case 1.1: *$\exists t_i \in \mathrm{stubborn}(m, q)$:*

Assume there is a transition t_i with $t_i \in \mathrm{stubborn}(m, q)$. We choose the smallest such i, which means that $m \xrightarrow{t_0 \ldots t_i} m_i$ with $t_i \in \mathrm{stubborn}(m, q)$ and $t_0, \ldots, t_{i-1} \notin \mathrm{stubborn}(m, q)$. Using requirement 1 (COM) we switch t_i to the front of the path and it holds that $m \xrightarrow{t_i t_0 \ldots t_{i-1}} m_i$.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q)}_{\pi_1} \underbrace{\xrightarrow{t_i t_0 \ldots t_{i-1}} (m_i, q)}_{\pi_2}$$

The following two cases show that requirement 4 (NLG) implies that all transitions $t_0, \ldots, t_{i-1}$ are satisfying the retarding formula ψ.

Case 1.1.1: *t_i is semi-invisible with respect to ψ:*

Requirement 2 (S-INV) ensures, if t_i is semi-invisible regarding ψ, then all states along the new path $t_i t_0 \ldots t_{i-1}$ are satisfying ψ.

Case 1.1.2: *$m \not\models \psi$:*

Otherwise using requirement 2 all transitions $t_0, \ldots, t_{i-1}$ are semi-invisible regarding ψ and together with the fact that requirement 3 ensures that t_i satisfies ψ, all states along the new path $t_i t_0 \ldots t_{i-1}$ are satisfying ψ.

In both cases 1.1.1 and 1.1.2 the path π_1 is extended by one state, which has an infinite accepting continuation in $\mathcal{P}$.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q) \xrightarrow{t_i} (m', q)}_{\pi_1} \underbrace{\xrightarrow{t_0 \ldots t_{i-1}} (m_i, q)}_{\pi_2}$$

Case 1.2: *$\nexists t_i \in \mathrm{stubborn}(m, q)$:*

Assume there is no transition $t_i \in \mathrm{stubborn}(m, q)$. Since q is in the accepting set of B, requirement 5 (KEY) ensures that there exists a key-transition $t^* \in \mathrm{stubborn}(m, q)$, s.t. $m \xrightarrow{t^* t_0 \ldots}$ is executable in N.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q)}_{\pi_1} \underbrace{\xrightarrow{t^*} (m^*, q) \xrightarrow{t_0 \ldots}}_{\pi_2}$$

It follows the same argumentation as in cases 1.1.1 and 1.1.2. The following two cases show that requirement 4 (NLG) implies that $t_0, \ldots$ are all satisfying the retarding formula ψ.

Case 1.2.1: *t^* is semi-invisible with respect to ψ:*

Requirement 2 (S-INV) ensures, if t^* is semi-invisible regarding ψ, then all states along the new path $t^* t_0 \ldots$ are satisfying ψ.

Case 1.2.2: $m \not\models \psi$:

Otherwise using requirement 2 all transitions $t_0, \dots$ are semi-invisible regarding ψ and together with the fact that $t^* \in \mathrm{stubborn}(m, q)$ and requirement 3 ensure that t^* satisfies ψ, all states along the new path $t^* t_0 \dots$ are satisfying ψ.

In both cases 1.2.1 and 1.2.2 the path π_1 is extended by one state, which has an infinite accepting continuation in $\mathcal{P}$.

$$\underbrace{q_s \longrightarrow (m_s, q_s) \longrightarrow (m, q) \xrightarrow{t^*} (m', q)}_{\pi_1} \underbrace{\xrightarrow{t_0} (m_0, q) \xrightarrow{\cdots}}_{\pi_2}$$

This construction can now be repeated as often as necessary to get an infinite accepting path in $\underline{\mathcal{P}}$, contradicting the assumption, that π_1 is the longest executable prefix.

Case 2: $\exists n \in \mathbb{N}_0\ \exists (m_n, q_n) : q_n \neq q$:

There exists a state (m_n, q_n) which is leaving the current Büchi state, $q_n \neq q$. Let q_n the first such state, then it holds that $\forall i < n : q_i = q$ and $m_n \models \varphi_j$, for a $j \in [1, h]$. Requirement 4 (NLG) ensures that there exists a t_i where $i \in [0, n]$ with $t_i \in \mathrm{stubborn}(m, q)$. Using requirement 1 (COM) we switch t_i to the front of the path.

$$m \xrightarrow{t_i t_0 \dots t_{i-1}} m_i \xrightarrow{t_{i+1} \dots t_n} m_n$$

It follows the same argumentation as in cases 1.1.1 and 1.1.2. The following two cases show that requirement 4 (NLG) implies that all transitions $t_0, \dots, t_{i-1}$ are satisfying the retarding formula ψ.

Case 2.1: *t_i is semi-invisible with respect to ψ:*

Requirement 2 (S-INV) ensures, if t_i is semi-invisible regarding ψ, then all states along the new path $t_i t_0 \dots t_{i-1}$ are satisfying ψ.

Case 2.2: $m \not\models \psi$:

Otherwise using requirement 2 all transitions $t_0, \dots, t_{i-1}$ are semi-invisible regarding ψ and together with the fact that requirement 3 ensures that t_i satisfies ψ, all states along the new path $t_i t_0 \dots t_{i-1}$ are satisfying ψ.

Hence it holds, that the extension $m \xrightarrow{t_i t_0 \dots t_{i-1}} m_i \xrightarrow{t_{i+1} \dots t_n}$ is also an infinite accepting path in $\mathcal{P}$, with the same traversed sequence of Büchi states, where at least one more state is executable in $\underline{\mathcal{P}}$. This construction can be repeated at most n-times, to find an infinite accepting path in which all transitions $t_0 \dots t_n$ (possibly in a different order) are executable in $\underline{\mathcal{P}}$. Which means that the path leaves q and changes to q_n. For the new Büchi state q_n we can, according to case 1 or case 2, apply the construction again, s.t. an executable infinite accepting path in $\underline{\mathcal{P}}$ is formed, contradicting the initial assumption on the choice of π.

6 Comparison

We compare the new automata based approach with the conventional LTL$_{-\mathbf{X}}$ method. We demonstrate for each weakened requirement promising potential gains in reduction efficiency. For simplicity, we assume that the different formulas φ_i, which we want to verify, are already negated. Some of the examples and their descriptions are inspired by [6].

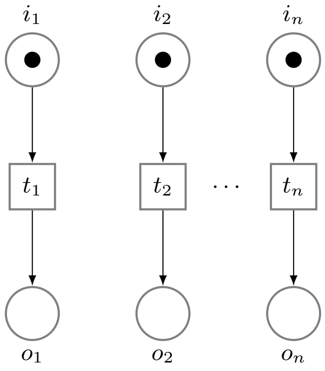


Fig. 5. A system modelled as P/T net with n concurrent processes.

The restriction to stuttering invariant formulas is dropped. A model checker, using the conventional LTL POR, had to turn off POR for those formulas that contain the $\mathbf{X}$-operator. It had to use standard brute force model checking in such situations. In the 2019 edition of the MCC [1] the $\mathbf{X}$-operator occurred in more than 25 % of the formulas in the LTL category. These formulas can now be verified with POR using the newly introduced method.

For the next examples consider the P/T net in Fig. 5. It models a system of $n \in \mathbb{N}$ concurrent processes, where each process has one transition t_j, with $1 \leq j \leq n$, one pre place i_j (input), and one post place o_j (output). The state space consists of 2^n markings with $2n2^n$ edges.

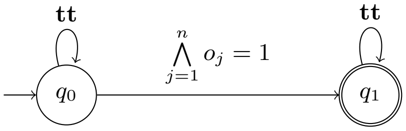


Fig. 6. Büchi automaton for LTL formula $\varphi_1 = \mathbf{F}\left(\bigwedge_{j=1}^{n} o_j = 1\right)$.

As first example we are considering the LTL formula $\varphi_1 = \mathbf{F}\left(\bigwedge_{j=1}^{n} o_j = 1\right)$. The corresponding Büchi automaton is shown in Fig. 6. The conventional visibility property based on definition 13 states that every transition is visible to φ_1, i.e., each transition t_j produces a token on o_j. This means, there is no reduction possible with the conventional LTL$_{-\mathbf{X}}$ POR and the reduced state space is the same as the original state space with 2^n markings. The product automaton has $2^n + 2$ reachable states: the initial state, 2^n states in Büchi state q_0, and one state in Büchi state q_1.

In the automata based approach every transition is semi-invisible in both Büchi states q_0 and q_1. The reason for this is that based on definition 18 we only care for the retarding formula and the retarding formula is always true. We see here the sizeable effect of the weakened visibility principle. In q_0 every singleton set $\{t_j\}$ is a valid stubborn set, as long as o_j does not contain a token. Without firing t_j the Büchi state q_0 cannot be left. And in addition to this, all other transitions are independent of each other. The reduced product automaton has $n+3$ reachable states: the initial state, $n+1$ chain-shaped states in Büchi state q_0, and one state in Büchi state q_1. The exponential reduction from 2^n+2 to $n+3$ states is based solely on the restriction of the visibility.

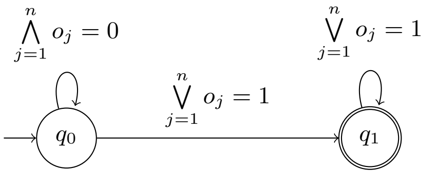

$\bigwedge_{j=1}^{n} o_j = 0$
$\bigvee_{j=1}^{n} o_j = 1$
$\bigvee_{j=1}^{n} o_j = 1$
q_0
q_1

Fig. 7. Büchi automaton for LTL formula $\varphi_2 = \left(\bigwedge_{j=1}^{n} o_j = 0\right) \mathbf{U} \left(\mathbf{G}\left(\bigvee_{j=1}^{n} o_j = 1\right)\right)$.

The second LTL formula $\varphi_2 = \left(\bigwedge_{j=1}^{n} o_j = 0\right) \mathbf{U} \left(\mathbf{G}\left(\bigvee_{j=1}^{n} o_j = 1\right)\right)$ we are considering, has the corresponding Büchi automaton shown in Fig. 7. Again, every transition in φ_2 is visible based on definition 13 and as such no reduction can be applied using the conventional method. The product automaton has 2^n+1 reachable states: the initial state, one state in Büchi state q_0, and 2^n-1 states in Büchi state q_1.

Using the new approach there is also no reduction possible in q_0 based on the semi-invisibility principle, since every transition can leave the current Büchi state. But we can reduce in q_1, due to fact that all transitions are semi-invisible. With this, every singleton stubborn set consisting of t_j is valid, as long as o_j is empty. The reduced product automaton has $\frac{n(n+1)}{2}+2$ reachable states: the initial state, one state in Büchi state q_0, and $\frac{n(n+1)}{2}$ states in Büchi state q_1. Due to the scope restriction of the semi-invisibility property, we only have a quadratic number of states instead of an exponential number of states.

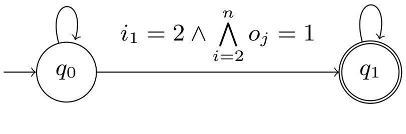
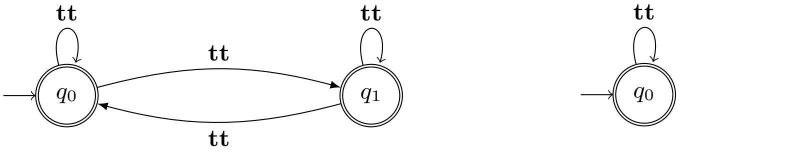


Fig. 8. Büchi automaton for LTL formula $\varphi_3 = \mathbf{F}\left(i_1 = 2 \wedge \bigwedge_{j=2}^{n} o_j = 1\right)$.

The LTL formula $\varphi_3 = \mathbf{F}\left(i_1 = 2 \wedge \bigwedge_{j=2}^{n} o_j = 1\right)$ is the third example we are considering and has the corresponding Büchi automaton shown in Fig. 8. The conventional definition of the visibility property (13) states that t_1 is invisible to φ_3 w.r.t. T and that all other transitions $t_2, \dots, t_n$ are visible to φ_3 w.r.t. the empty set. This means in the initial state we can reduce T to the valid stubborn set $\{t_1\}$. As consequence the state space is reduced to $2^{n-1} + 1$ markings. And the product automaton is also reduced to $2^{n-1} + 2$ reachable states: the initial state, and $2^{n-1} + 1$ states in Büchi state q_0. Note that in q_1 no state is reachable, since $i_1 = 2$ can never be satisfied.

With the automata based approach we can use requirement 5 (KEY) to get a substantially better reduction. Requirement 5 states that only in an accepting Büchi state an activated transition has to be in the stubborn set. We use the fact that q_0 is not an accepting state and with this the empty set is a valid stubborn set. Although there are activated transitions in the initial state, with none of them it is possible to leave Büchi state q_0. As mentioned before, the reason for this is that $i_1 = 2$ can never be satisfied. Independent from n, the reduced product automaton has always two states: the initial state, and one state in q_0. This extreme reduction power from $2^{n-1} + 2$ to 2 states is based on the requirement, that the KEY principle only has to hold in accepting states.



(a) Unfavourable Büchi automaton

(b) Simplified Büchi automaton

Fig. 9. Büchi automata for LTL formula $\varphi_4 = \mathbf{false}$.

One thing that should be mentioned is that the reduction power strongly dependents on the formula and the used Büchi automaton. With some effort we can construct unfavorable Büchi automata where the conventional LTL$_{-\mathbf{X}}$ POR performs better or equally good. Fig. 9 (a) shows such an unfavorable Büchi automaton for the LTL formula $\varphi_4 = \mathbf{false}$. It accepts every path. The problem is that in every arbitrary state of the product automaton it is possible to switch between the Büchi states and with this, no reduction can be applied based on the retarding formula. Although every transition is invisible and thus independent of other transitions, we cannot exploit that in this case.

The product automaton with conventional POR has a size of $2n + 3$ states, while the product automaton built with the new approach has $2 \cdot 2^n + 1$ states. The good news is, the LTL formula $\varphi_4 = \mathbf{false}$ can also be represented with only one Büchi state, where simply one Büchi state from Fig. 9 (a) is removed, resulting in Fig. 9 (b). With this Büchi automaton the problem vanishes and the reduction power is identical for methods. Both have $n + 2$ states in the product automaton. We conclude that the newly introduced POR can substantially increase the reduction efficiency. Table 1 summarizes the advantages and disadvantages once again.

Property/principle	**Old**	**New**
stubborn set generator	$M \to 2^T$	$M \times Q \to 2^T$
$\mathbf{X}$-operator	not supported	improved: supported
VIS	✓	improved: replaced by weaker S-INV
IGN	✓	improved: not required
KEY	✓	improved: required only in accepting states
NLG	not required	✓

Table 1. Comparison of the old and the new method.

7 Related work and conclusion

Most POR approaches for LTL are focusing on the reduction of the transition system and omitting the additional information available from the Büchi automaton. To the best of our knowledge, there exist two other approaches that uses information from the Büchi state. First, in [11] the original invisibility property was relaxed to those propositions that lie ahead of the current Büchi state. And second, in [8] the invisibility property was further restricted to only one propositions at a time, and only for retarding formulas. They further introduced the idea to first build the product automaton with the original state space and then reduce the product automaton, using the additional information available from the Büchi automaton. The drawback in [8] is that it only works on elementary Büchi automata.

Our contribution is a generalization and extension of the ideas presented in [6] and [8]. We introduced a new semi-invisibility principle which also reduces the scope of the formula to one proposition at a time, namely to the retarding formula of the current Büchi state. Furthermore, the new semi-invisibility principle only has to hold in one direction. We also introduced a new principle, called NLG, for the reduction: all transition sequences, which do not use transitions from the stubborn set, cannot leave the current Büchi state.

With this, some requirements can be weakened or even dropped, resulting in several advantages over the conventional POR. One drawback of the conventional approach is that it only works on stuttering invariant formulas, that is, formulas which are not containing the $\mathbf{X}$-operator. In [5] a POR approach was introduced which is able to do POR with the $\mathbf{X}$-operator. They use a heuristic to detect and extend stuttering invariant components of the Büchi automaton while translating the LTL formula to a Büchi automaton. These components are then used to guide the reduction of the state space. By contrast, in our work the reduction is only applied within a single Büchi state and thus, stuttering becomes irrelevant. Finite stuttering within a Büchi state does not change the accepting behavior, and infinite stuttering can always be avoided, if this was possible in the original product automaton, due to the NLG principle of the reduction. With this, we can also drop the restriction to LTL$_{-\mathbf{X}}$. Our method is able to verify the full class of LTL. Since stuttering is irrelevant the non-ignorance principle can also be dropped. Furthermore the key-transition principle can be reduced to states which are in the accepting set.

All in all we have weakened or dropped several requirements compared to other POR methods. We showed with examples, which involved some of the weakened or dropped requirements, the reduction efficiency. Compared to conventional methods the reduction power can be extremely better, e.g., $n+3$ vs. 2^n+2 states, or 2 vs. $2^{n-1}+2$ states. The reduction power depends on the formula, or to be exact, on its corresponding Büchi automaton.

In future works we want to expand our new method to other formalisms than P/T nets. Furthermore, we want to explore whether the new automata based approach can be tuned to be always better or at least equally good as the conventional POR. In addition to this, we want to implement our new method in our explicit model checker LoLA 2 [24]. LoLA 2 has won the LTL category in the yearly MCC [1] for the last 3 years, 2017 - 2019. We expect a substantial performance increase with the new automata based POR. Empirical investigation will show the true power of this approach, and if it will perform better on relevant systems.

References

1. F. K. et al. Presentation of the 9th edition of the model checking contest. In *Proc. TACAS, LNCS 11429*, pages 50–68, 2019.
2. R. Gerth, R. Kuiper, D. A. Peled, and W. Penczek. A partial order approach to branching time logic model checking. *Inf. Comput.*, 150(2):132–152, 1999.
3. P. Godefroid and P. Wolper. A partial approach to model checking. volume 110, pages 305–326, 1994.
4. J. F. Jensen, T. Nielsen, L. K. Oestergaard, and J. Srba. TAPAAL and reachability analysis of P/T nets. *ToPNoC*, 11, LNCS 9930:307–318, 2016.
5. S. Kan, Z. Huang, Z. Chen, W. Li, and Y. Huang. Partial order reduction for checking LTL formulae with the next-time operator. *J. Log. Comput.*, 27(4):1095–1131, 2017.
6. C. Koch. Weiterentwicklung von Methoden der Partial Order Reduction. Master thesis, Universität Rostock, 2013.
7. L. M. Kristensen, K. Schmidt, and A. Valmari. Question-guided stubborn set methods for state properties. *Formal Methods in System Design*, 29(3):215–251, 2006.
8. A. Lehmann, N. Lohmann, and K. Wolf. Stubborn sets for simple linear time properties. In *Proc. PETRI NETS, LNCS 7347*, pages 228–247, 2012.
9. T. Liebke and K. Wolf. Taking some burden off an explicit CTL model checker. In *Proc. PETRI NETS, LNCS 11522*, pages 321–341, 2019.
10. D. A. Peled. All from one, one for all: on model checking using representatives. In *Proc. CAV, LNCS 697*, pages 409–423, 1993.
11. D. A. Peled, A. Valmari, and I. Kokkarinen. Relaxed visibility enhances partial order reduction. *Formal Methods in System Design*, 19(3):275–289, 2001.
12. D. A. Peled and T. Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Inf. Process. Lett.*, 63(5):243–246, 1997.
13. K. Schmidt. Stubborn sets for standard properties. In *Proc. ICATPN, LNCS 1639*, pages 46–65, 1999.

14. K. Schmidt. How to calculate symmetries of Petri nets. *Acta Inf.*, 36(7):545–590, 2000.
15. A. Valmari. Error detetction by reduced reachability graph generation. *9th European Workshop on Application and Theory of Petri nets, Venice, Italy*, pages 95–112, 1988.
16. A. Valmari. Eliminating redundant interleavings during concurrent program verification. In *Proc. PARLE, LNCS 366*, pages 89–103, 1989.
17. A. Valmari. Stubborn sets for reduced state space generation. In G. Rozenberg, editor, *Proc. PETRI NETS, LNCS 483*, pages 491–515, 1989.
18. A. Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1(4):297–322, 1992.
19. A. Valmari. The state explosion problem. In *Lectures on Petri Nets I:,LNCS 1491*, pages 429–528, 1996.
20. A. Valmari. Stubborn set methods for process algebras. In *Proc. DIMACS Workshop on Partial Order Methods in Verification*, volume 29, page 213âĂŞ231, 1997.
21. A. Valmari and H. Hansen. Stubborn set intuition explained. *T. Petri Nets and Other Models of Concurrency*, 12:140–165, 2017.
22. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. (LICS*, pages 332–344, 1986.
23. K. Varpaaniemi. On the stubborn set method in reduced state space generation. Technical report, 1998.
24. K. Wolf. Petri net model checking with LoLA 2. In *Proc. PETRI NETS, LNCS 10877*, pages 351–362, 2018.