Token Elimination in Model Checking of Petri Nets*

Nicolaj Ø. Jensen, Kim G. Larsen, and Jiří Srba

Department of Computer Science, Aalborg University
Selma Lagerløfs Vej 300, 9220 Aalborg, Denmark
{noje,kgl,srba}@cs.aau.dk

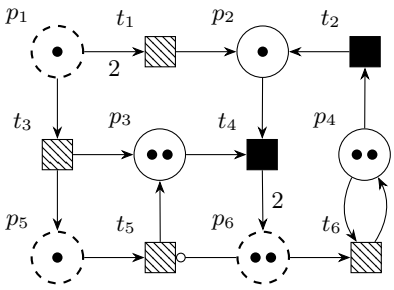
Abstract. We propose a novel state-space reduction framework to improve the performance of model checking of Petri nets. We provide two instances of the framework: a static technique that considers only the structure of the net, and a dynamic technique that additionally considers the current marking. By analyzing impossible, visible, and directional effects of transitions, we identify places where tokens can be removed while preserving the property in question. Unlike structural reductions, our techniques modify only the current marking, allowing the net structure to be reused in multiple subproblems concurrently, which can be beneficial for example for CTL model checking. We prove the correctness of our techniques and implement them in the open-source tool TAPAAL, a repeated winner in the CTL category in the annual model checking contest (MCC). We measure our methods' performance on the MCC 2023 benchmark using the CTL categories and demonstrate that our methods reduce time and, especially, memory usage. Our dynamic method explores 39.3% fewer configurations on average and achieves two orders of magnitude speedup on at least one query on 23.7% of non-trivial models.

Keywords: Petri Nets · State-Space Reduction · Model Checking.

1 Introduction

The main obstacle of verifying properties of concurrent systems is the state-space explosion problem [8] that follows from the many possible interleavings of even simple subsystems and their composed intermediate states when run in parallel. The state space of a concurrent system is often exponentially larger than any of its constituent systems, and exploring every state to verify a property is intractable in practice. Therefore, many approaches have been developed to simplify systems structurally [7,8,22,26], to disregard equivalent traces through partial-order reductions [7,8,14,24,27], and to symbolically verify multiple states simultaneously [8,22,23]. We present a reduction that combines ideas from all the aforementioned methods to simplify the states and the amount of branching in the reduced system. The reduction is presented in the context of the weighted Petri net formalism with inhibitor arcs.

* Funded by the VILLUM INVESTIGATOR project S4OS.



$\varphi := p_3 = 0$
$\text{places}_{\Uparrow}(\varphi) = \emptyset$
$\text{places}_{\Downarrow}(\varphi) = \{p_3\}$
$C^M_{\Uparrow} = P \setminus \{p_1, p_4\}$
$C^M_{\Downarrow} = P$
$C^M_F = T \setminus \{t_1\}$
$D^{\varphi,M}_{\Uparrow} = \{p_2, p_4\}$
$D^{\varphi,M}_{\Downarrow} = \{p_3\}$
$T^{\varphi,M} = \{t_2, t_4\}$
$P^{\varphi,M} = \{p_2, p_3, p_4\}$

Fig. 1: Example Petri net and sets computed by our technique for the depicted marking M and property φ. Solid transitions are in $T^{\varphi,M}$ and are the only ones that require firing to eventually satisfy φ from the given marking, if possible. Dashed places are not in $P^{\varphi,M}$ and can have their tokens removed without affecting whether EFφ holds. Arcs without weight annotation have weight 1.

Petri nets are directed bipartite graphs consisting of places (circles) and transitions (squares). Places can contain a number of tokens, and the token configuration across all places forms a state (a marking) of the Petri net. For a transition to fire, one or more tokens are required in each of its predecessor places, as firing it will remove a token from each predecessor place and add a token to each successor place, resulting in a new marking. In a weighted Petri net, the arcs are weighted, affecting how many tokens are removed and added. Inhibitor arcs, depicted with circle-headed arrows, are a special type of arc that prevent the transition at their head from firing if the number of tokens at the source place of the arc equals or exceeds the arc's weight. Petri nets are well suited for modeling parallel systems, where various discrete resources move from state to state, for example, packets in a network or tasks in a workflow system.

Consider the example Petri net illustrated in Figure 1 and let M denote the depicted marking. Now assume that we want to check whether it is possible to reach a marking with 0 tokens in place p_3, formally whether $M \vDash \text{EF}\varphi$ holds where $\varphi \equiv (p_3 = 0)$. It is possible to show that this property holds in multiple ways, e.g. firing the sequence $t_2t_4t_4$ will reach such a marking. However, some transitions and some tokens are unnecessary to demonstrate that the property holds. For example, the transition t_6 can always be omitted in such witness traces, and the number of tokens in p_6 is irrelevant since none of the shortest witness traces (those consisting of the fewest firings, i.e. $t_2t_4t_4$ and $t_4t_2t_4$) will not consume from p_6 anyway. Our approach in this paper computes a set of transitions $T^{\varphi,M}$ (solid squares in the example) such that if a trace to a φ-satisfying marking exists, then the shortest traces to such a marking consists exclusively of firings from this set. We use this set to identify a set of places $P^{\varphi,M}$, such that if marking M' matches M in these places but has 0 tokens elsewhere (elsewhere are dashed circles in the example), then it is guaranteed

that the shortest witness traces are preserved, and thus $M' \vDash \mathrm{EF}\varphi$ if and only if $M \vDash \mathrm{EF}\varphi$.

Removing tokens from the reached markings during a state-space search has two major benefits: (i) two different markings where unimportant tokens are removed can become identical and this helps to reduce the size of the state space, and (ii) removing unimportant tokens disables unimportant transitions and hence reduces the branching degree of the state space.[1]

Our contributions. We present a novel on-the-fly technique that reduces the reachable state space by removing tokens from the explored markings. As a direct consequence of only modifying markings, our method is suitable where other techniques are not, e.g. in on-the-fly model checking. Our technique is also applicable to unrestricted (even unbounded) Petri nets—all theorems are proven in full generality. The termination of the state-space search is, of course, not guaranteed in such a case, but like many other reductions, our technique will reduce some infinite state spaces to finite ones.

We present two versions of our technique. The static version uses the structure of the net to compute visible effects, distinguishing between visible increases and decreases, and thus determines where tokens are invisible and can be removed. When using the static version, the set of visible places can be cached to reduce overhead. The dynamic version is a refinement that first, based on the current marking, approximates a set of transitions that can never be enabled in future markings. This allows us to disregard some impossible effects that would otherwise be structurally visible, limiting how visibility propagates in the net and resulting in more places where tokens can be removed.

Our techniques only preserve reachability properties, but due to their ability to be used in concurrent model checking, our experiments focus on Computation Tree Logic (CTL) [8] where reachability sub-properties often must be checked, potentially with many different initial markings. Structural reductions that only preserve reachability properties cannot be used in this context, whereas our reduction can. We implement our techniques in the popular model checker Tapaal, and evaluate it using the Model Checking Contest 2023 [18] benchmark with a focus on challenging queries that challenge the state-of-the-art tool. We show that we save both memory and time when implemented in the CTL algorithm of Tapaal [10].

Related work Partial-order reductions include many similar techniques such as ample sets [17,24], persistent sets [14], and stubborn sets [14,27]. Each preserves different kinds of properties, but all share the idea that not all interleavings need to be searched. Recently, stubborn reductions have been made available for reachability games [5] and systems with time [3,4]. Our technique shares similarities with identification of stubborn sets [7,14,19]. However, our technique

[1] Removing tokens that inhibit transitions may result in more behavior, but inhibitor arcs are often few, and in the rare cases where our technique uninhibits transitions, they are always excluded through stubborn reductions anyway.

is coarser in some ways, as it (i) is based on visible effects instead of just visible transitions, (ii) limits visibility propagation by considering impossible transition firings and effects, and (iii) is only guaranteed to preserve the shortest traces to satisfying markings. Moreover, we simplify the marking rather than the set of considered transitions. In our early experiments, we applied the coarser marking-dependent *up-set* from [19] when determining which places in the query needed change. However, it did not provide any positive effect on the benchmark.

Structural reductions of Petri nets [7,22,26] have been around almost since the introduction of Petri nets. Structural reductions modify the Petri net itself [22,26]. Many possible transformations exist, some even incorporate SMT solvers [26]. Our technique is comparable to a combination of structural reduction rules I and M found in Tapaal [7]. However, in addition to the refinements (i)-(iii) mentioned earlier, we modify only the marking instead of the net structure. We are aware of only one structural reduction that does so, and it simulates firing transitions in the initial marking [26]. The fact that we only modify the marking allows us to use our technique on-the-fly without allocating memory for several reduced versions of the Petri net.

CTL is a branching-time temporal logic used to specify properties of concurrent systems [8]. It allows for the expression of temporal properties by quantifying over paths in a system's state-transition graph. Note that verifying CTL properties of Petri nets even without inhibitor arcs is generally undecidable [6] and reachability problems have recently been proved to be Ackermann-complete [9]. These undecidability/complexity results do not affect the correctness of our techniques, and the technique can be applied to any reachability problem directly or to subproblems of CTL about reachability. In [20], the authors also explore subproblem-specific reductions in CTL model checking [20], however, the way we reduce markings by token elimination has not been studied before.

Structure of the Paper. In Section 2 we present the Petri net formalism and the reachability problem. In Section 3 we present our techniques and their correctness. Our experiments and results are shown in Section 4, and we conclude on our findings in Section 5.

2 Petri Nets and the Reachability Problem

We shall first introduce the necessary definitions.

Definition 1 (Transition System). *Given a set Π of atomic propositions, a* transition system (TS) *over Π is a 4-tuple $\mathcal{T} = \langle \mathcal{S}, \mathcal{A}, \rightarrow, L \rangle$ such that $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a finite set of actions, $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a transition relation, and $L : \mathcal{S} \rightarrow 2^{\Pi}$ is a labeling function assigning each state to a set of propositions.*

We write $s \xrightarrow{a} s'$ whenever $\langle s, a, s' \rangle \in \rightarrow$. We write $s \xrightarrow{a}$ whenever $\langle s, a, s' \rangle \in \rightarrow$ for some $s' \in \mathcal{S}$ and say that a is *enabled* in s. We write $s \rightarrow s'$ if $\langle s, a, s' \rangle \in \rightarrow$ for some $a \in \mathcal{A}$ and write $s \not\rightarrow$ if $\langle s, a, s' \rangle \notin \rightarrow$ for any $a \in \mathcal{A}, s' \in \mathcal{S}$. We extend the notion $\xrightarrow{a}$ inductively to traces $w \in \mathcal{A}^*$ such that $s \xrightarrow{\epsilon} s$ for all $s \in \mathcal{S}$ and

$s \xrightarrow{wa} s'$ if $s \xrightarrow{w} s''$ and $s'' \xrightarrow{a} s'$. Finally, $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$.

Definition 2 (Reachability and State Properties). *Given a set Π of atomic propositions,* $\mathrm{EF}\varphi$ *is a* reachability property *where* state property φ *(in negation normal form) is defined inductively by the grammar $\varphi ::= \pi \mid \neg\pi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$ where $\pi \in \Pi$. Let Φ be the set of all state properties in negation normal form.*

We assume standard semantics and write $s \vDash \varphi$ if the state s satisfies φ, and we write $s \vDash \mathrm{EF}\varphi$ if $s \rightarrow^* s'$ and $s' \vDash \varphi$. The reachability problem involves checking whether a reachability property holds in a given state. For this paper, it is necessary to know that reachability properties are a strict subset of the branching-time logic known as computation tree logic (CTL) but we refer to [8] for further details on CTL.

Definition 3 (Petri Net with Inhibitor Arcs). *A* Petri net with inhibitor arcs (PN) *is a 4-tuple $N = \langle P, T, W, I\rangle$ where*

- *P is a finite set of* places*,*
- *T is a finite set of* transitions *such that $P \cap T = \emptyset$,*
- *$W : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is an incidence matrix, and*
- *$I : P \times T \to \mathbb{N} \cup \{\infty\}$ is an inhibitor weight matrix.*

Definition 4 (Marking). *For a PN N with places P, a marking $M : P \to \mathbb{N}_0$ is a function that assigns to each place a number of tokens present in the place. The set $\mathcal{M}(N)$ is the set of all markings of PN N.*

The effect matrix $E : T \times P \to \mathbb{Z}$ is a function defined as $E(t,p) = W(t,p) - W(p,t)$. We also use the following notation where $p \in P$ and $t \in T$:

$^\bullet p = \{t' \in T \mid W(t',p) > 0\}$ is the preset of p,
$p^\bullet = \{t' \in T \mid W(p,t') > 0\}$ is the postset of p,
$^\bullet t = \{p' \in P \mid W(p',t) > 0\}$ is the preset of t,
$t^\bullet = \{p' \in P \mid W(t,p') > 0\}$ is the postset of t,
$^+p = \{t' \in {}^\bullet p \mid E(t',p) > 0\}$ is the increasing preset of p,
$p^- = \{t' \in p^\bullet \mid E(t',p) < 0\}$ is the decreasing postset of p,
$t^+ = \{p' \in t^\bullet \mid E(t,p') > 0\}$ is the increased postset of t,
$^-t = \{p' \in {}^\bullet t \mid E(t,p') < 0\}$ is the decreased preset of t,
$p^\circ = \{t' \in T \mid I(p,t') \neq \infty\}$ is the inhibited postset of p,
$^\circ t = \{p' \in P \mid I(p',t) \neq \infty\}$ is the inhibiting preset of t.

We extend the notation to sets such that for a set X of places or transitions, $^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and likewise for the other preset and postset operators.

For PNs, without loss of generality, we consider atomic proposition $\pi \in \Pi$ to be of the form $k \leq \sum_p v_p p$ where $k, v_p \in \mathbb{Z}$. Marking M satisfies atomic proposition π, written $M \vDash \pi$, if the inequality holds when all occurrences of places $p \in P$ in π are replaced with $M(p)$. For example, if $p_1, p_2, p_3 \in P$ then

EF $3 \leq p_1 \wedge 1 \leq p_2 - p_3$ asserts that it is possible to reach a marking, where p_1 contains at least 3 tokens and there is at least 1 more token in p_2 than in p_3. The negation of π is achieved by negating all weights v_p and replacing k with $-k+1$.

A PN N defines a TS $\mathcal{T}(N) = \langle \mathcal{S}, \mathcal{A}, \rightarrow, L \rangle$ over Π where $S = \mathcal{M}(N)$, $\mathcal{A} = T$, and $M \xrightarrow{t} M'$ if for all $p \in P$ we have $M(p) \geq W(p,t)$ and $M(p) < I(p,t)$ and $M'(p) = M(p) + E(t,p)$. Finally, $\pi \in L(M)$ iff $M \vDash \pi$ where $\pi \in \Pi$.

Definition 5 (Increasing and Decreasing Support). *The* increasing support *is a function* $\text{places}_{\Uparrow} : \Phi \to 2^P$ *defined inductively as:*

$$\begin{aligned} \text{places}_{\Uparrow}(k \leq \sum_p v_p p) &= \{p \in P \mid v_p > 0\} \\ \text{places}_{\Uparrow}(\varphi_1 \wedge \varphi_2) &= \text{places}_{\Uparrow}(\varphi_1) \cup \text{places}_{\Uparrow}(\varphi_2) \\ \text{places}_{\Uparrow}(\varphi_1 \vee \varphi_2) &= \text{places}_{\Uparrow}(\varphi_1) \cup \text{places}_{\Uparrow}(\varphi_2) \ . \end{aligned}$$

The decreasing support *is a function* $\text{places}_{\Downarrow} : \Phi \to 2^P$ *defined inductively as:*

$$\begin{aligned} \text{places}_{\Downarrow}(k \leq \sum_p v_p p) &= \{p \in P \mid v_p < 0\} \\ \text{places}_{\Downarrow}(\varphi_1 \wedge \varphi_2) &= \text{places}_{\Downarrow}(\varphi_1) \cup \text{places}_{\Downarrow}(\varphi_2) \\ \text{places}_{\Downarrow}(\varphi_1 \vee \varphi_2) &= \text{places}_{\Downarrow}(\varphi_1) \cup \text{places}_{\Downarrow}(\varphi_2) \ . \end{aligned}$$

Finally, we let $\text{places}(\varphi) = \text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)$ *be the* support *of the state property* φ.

The intuition is that $\text{places}_{\Uparrow}(\varphi)$ (resp. $\text{places}_{\Downarrow}(\varphi)$) denote the set of places where more (resp. fewer) tokens are potentially needed for φ to be satisfied if not already satisfied. The set $\text{places}(\varphi)$ is also said to be *directly visible* to φ, because it is not possible to go from a marking where φ is not satisfied to one where φ is satisfied without firing a transitions that affects one of these places in the appropriate direction. Formally:

Lemma 1. *Let* $N = \langle P, T, W, I \rangle$ *be a PN,* $M, M' \in \mathcal{M}(N)$ *markings,* φ *a state property, and* $t \notin {}^{+}\text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)^{-}$ *a transition. If* $M \not\vDash \varphi$ *and* $M \xrightarrow{t} M'$*, then* $M' \not\vDash \varphi$.

3 Token Elimination

We now introduce our token elimination technique, which aims to identify (in a dynamic or static way) a set of places where the number of tokens in any reachable marking can be reset to zero while preserving a given reachability property.

We say that elements (places, transitions, tokens, or effects) are *invisible* if they do not affect whether the reachability property holds, i.e. our techniques identify invisible tokens. They do so by computing an over-approximation of the visible transitions and visible effects. Remark that we will often describe the content of our over-approximation as *visible* even though some of it may be *invisible* in practice. We first present the static technique focusing on intuition and then the dynamic technique with full proofs (of which static is a simplified case). For the rest of this section, let us fix a PN $N = \langle P, T, W, I \rangle$ and reachability property EFφ.

3.1 Static Token Elimination

Our static token-elimination technique computes two sets of places: $S^{\varphi}_{\Uparrow}$ and $S^{\varphi}_{\Downarrow}$ that exhibit the following properties. If $M, M' \in \mathcal{M}(N)$ are markings such that $M \to^* M'$ and $M' \vDash \varphi$ and $w \in T^*$ is one of the shortest sequences of transitions such that $M \xrightarrow{w} M'$ (there can be more than one such sequence) then:

- if $p \in \text{places}_{\Uparrow}(\varphi)$ or there exists a transition t in w that consumes from p, then $p \in S^{\varphi}_{\Uparrow}$, and
- if $p \in \text{places}_{\Downarrow}(\varphi)$ or there exists a transition t in w that is inhibited by p, then $p \in S^{\varphi}_{\Downarrow}$.

In other words, $S^{\varphi}_{\Uparrow}$ is a superset of places where we need more tokens and positive effects are visible. Similarly, $S^{\varphi}_{\Downarrow}$ is a superset of places where we need fewer tokens and where negative effects are visible. We compute $S^{\varphi}_{\Uparrow}$ and $S^{\varphi}_{\Downarrow}$ inductively as the least fixed point of the following constraints:

(S1) if $p \in \text{places}_{\Uparrow}(\varphi)$ then $p \in S^{\varphi}_{\Uparrow}$,
(S2) if $p \in \text{places}_{\Downarrow}(\varphi)$ then $p \in S^{\varphi}_{\Downarrow}$,
(S3) if $p \in S^{\varphi}_{\Uparrow}$ then ${}^{\bullet+}p \subseteq S^{\varphi}_{\Uparrow}$,
(S4) if $p \in S^{\varphi}_{\Uparrow}$ then ${}^{\circ+}p \subseteq S^{\varphi}_{\Downarrow}$,
(S5) if $p \in S^{\varphi}_{\Downarrow}$ then $({}^{\bullet}(p^-) \setminus \{p\}) \subseteq S^{\varphi}_{\Uparrow}$,
(S6) if $p \in S^{\varphi}_{\Downarrow}$ then ${}^{\circ}(p^-) \subseteq S^{\varphi}_{\Downarrow}$,
(S7) if $p \in S^{\varphi}_{\Downarrow}$ and $\exists t \in p^-.W(p,t) > 1$ then $p \in S^{\varphi}_{\Uparrow}$.

We briefly give the intuition for these constraints:

- **(S1)**, **(S2)**: If a place is in $\text{places}_{\Uparrow}(\varphi)$ or $\text{places}_{\Downarrow}(\varphi)$, then effects of the appropriate direction are (directly) visible to the satisfaction of φ.
- **(S3)**, **(S4)**: If increases are visible in p, then any transition t with positive effect on p is visible, and hence effects that enable t are also visible. That is, increases in the preset of the positive preset of p are visible, and decreases in the inhibiting preset of the positive preset of p are visible.
- **(S5)**, **(S6)**, **(S7)**: Similarly, if decreases are visible in p, then any transition t with negative effect on p is visible, and hence effects that enable t are also visible. However, since we are interested in decreases to p, adding tokens to

p would be counter-productive, which is why we do not require p in $S^{\varphi}_{\Uparrow}$ in **(S5)**. However, this logic does not hold if some $t \in p^-$ must consume more than one token from p to fire, in which case we may have to add tokens to p before we can empty it with t. Hence, **(S7)**.

Finally, let $T^{\varphi}_{\text{sta}} = {}^+(S^{\varphi}_{\Uparrow}) \cup (S^{\varphi}_{\Downarrow})^-$ be the set of transitions that have a visible effect, and let $P^{\varphi}_{\text{sta}} = \text{places}(\varphi) \cup {}^\bullet T^{\varphi}_{\text{sta}} \cup {}^\circ T^{\varphi}_{\text{sta}}$ be the set of places with visible tokens. As we shall prove for the corresponding dynamic version of T^{φ}_{sta}, the shortest trace to a marking satisfying φ does not include transitions outside of this set. Therefore, it does not matter for the satisfaction of $\text{EF}\varphi$ if we modify the marking in a way that disables transitions not in T^{φ}_{sta}. Hence, we define a reduction that removes these invisible tokens.

Definition 6 (Static Token Elimination). *The* static token elimination *abstraction* $\alpha^{\varphi}_{\text{sta}} : \mathcal{M}(N) \to \mathcal{M}(N)$ *is given by*

$$\alpha^{\varphi}_{\text{sta}}(M)(p) = \begin{cases} M(p) & \text{if } p \in P^{\varphi}_{\text{sta}} \\ 0 & \text{otherwise .} \end{cases}$$

Theorem 1 ($\alpha^{\varphi}_{\text{sta}}$ Preserves Reachability). *Let M be a marking and $\text{EF}\varphi$ a reachability property. We have $M \vDash \text{EF}\varphi$ iff $\alpha^{\varphi}_{\text{sta}}(M) \vDash \text{EF}\varphi$.*

We note that P^{φ}_{sta} depends only on the structure of the net and φ. Hence, it can be cached based on φ if it is a reoccurring subproblem on the same net. We take advantage of this in our CTL algorithm implementation.

3.2 Dynamic Token Elimination

The dynamic token elimination is a refinement of the static technique. In addition to visible effects, we shall now also consider whether the effects are possible from the current marking. Let us fix a current marking M. The set $C^M_{\Uparrow}$ (resp. $C^M_{\Downarrow}$) is a set of places that may potentially have their number of tokens increased (resp. decreased) in a future marking. Additionally, the set C^M_F contains transitions that may be enabled in a future marking. We define $C^M_{\Uparrow}$, $C^M_{\Downarrow}$, and C^M_F inductively as the least fixed point of the following constraint:

(C) If $t \in T$ and for all $p \in P$, we have $(W(p,t) \leq M(p)$ or $p \in C^M_{\Uparrow})$ and $(I(p,t) > M(p)$ or $p \in C^M_{\Downarrow})$, then $t \in C^M_F$ and $t^+ \subseteq C^M_{\Uparrow}$ and ${}^-t \subseteq C^M_{\Downarrow}$.

We now state the properties of the sets defined by this constraint.

Lemma 2. *Let $p \in P$ be a place and $t \in T$ be a transition.*

- *If $p \notin C^M_{\Uparrow}$ then $M'(p) \leq M(p)$ for all M' s.t. $M \to^* M'$,*
- *If $p \notin C^M_{\Downarrow}$ then $M'(p) \geq M(p)$ for all M' s.t. $M \to^* M'$,*
- *If $t \notin C^M_F$ then $M' \overset{t}{\not\to}$ for all M' s.t. $M \to^* M'$.*

We can now use $C^M_{\Uparrow}$, $C^M_{\Downarrow}$, and C^M_F to find refinements of $S^\varphi_{\Uparrow}$ and $S^\varphi_{\Downarrow}$. The refined sets $D^{\varphi,M}_{\Uparrow}$ and $D^{\varphi,M}_{\Downarrow}$ contain places where positive and negative effects, respectively, are possible and visible. We define $D^{\varphi,M}_{\Uparrow}$ and $D^{\varphi,M}_{\Downarrow}$ inductively as the least fixed point of the following constraints:

(D1) if $p \in \text{places}_{\Uparrow}(\varphi) \cap C^M_{\Uparrow}$ then $p \in D^{\varphi,M}_{\Uparrow}$
(D2) if $p \in \text{places}_{\Downarrow}(\varphi) \cap C^M_{\Downarrow}$ then $p \in D^{\varphi,M}_{\Downarrow}$
(D3) if $p \in D^{\varphi,M}_{\Uparrow}$ then $({}^\bullet({}^+p \cap C^M_F) \cap C^M_{\Uparrow}) \subseteq D^{\varphi,M}_{\Uparrow}$
(D4) if $p \in D^{\varphi,M}_{\Uparrow}$ then $({}^\circ({}^+p \cap C^M_F) \cap C^M_{\Downarrow}) \subseteq D^{\varphi,M}_{\Downarrow}$
(D5) if $p \in D^{\varphi,M}_{\Downarrow}$ then $({}^\bullet(p^- \cap C^M_F) \cap (C^M_{\Uparrow} \setminus \{p\})) \subseteq D^{\varphi,M}_{\Uparrow}$
(D6) if $p \in D^{\varphi,M}_{\Downarrow}$ then $({}^\circ(p^- \cap C^M_F) \cap C^M_{\Downarrow}) \subseteq D^{\varphi,M}_{\Downarrow}$
(D7) if $p \in D^{\varphi,M}_{\Downarrow}$ and $\exists t \in p^- \cap C^M_F . W(p,t) > 1$ and $p \in C^M_{\Uparrow}$ then $p \in D^{\varphi,M}_{\Uparrow}$

The constraints correspond to **(S1)**-**(S7)**, but propagation is now limited using $C^M_{\Uparrow}$, $C^M_{\Downarrow}$, and C^M_F. Next, let

$$T^{\varphi,M}_{\text{dyn}} = \left({}^+(D^{\varphi,M}_{\Uparrow}) \cup (D^{\varphi,M}_{\Downarrow})^-\right) \cap C^M_F \tag{1}$$

be the set of fireable transitions with visible effects.

Lemma 3. *Let $M \to^* M'$ such that $M' \vDash \varphi$. If $w \in T^*$ is some shortest sequence of transitions (there can be more than one) such that $M \xrightarrow{w} M'$ then necessarily $w \in (T^{\varphi,M}_{\text{dyn}})^*$.*

Proof. Let $w = t_1 \dots t_n \in T^*$ be some shortest sequence such that $M \xrightarrow{w} M'$ with $M' \vDash \varphi$. If $n = 0$, then $M = M'$ and the claim holds trivially. If $n = 1$ and hence $w = t \in T$, then $M \nvDash \varphi$, otherwise w would be shorter. By contraposition of Lemma 1, we have that $t \in {}^+\text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)^-$, otherwise it is not possible that $M' \vDash \varphi$. Since t is enabled, we know $t \in C^M_F$. By **(D1)**, we know $\text{places}_{\Uparrow}(\varphi) \subseteq D^{\varphi,M}_{\Uparrow}$, so if $t \in {}^+\text{places}_{\Uparrow}(\varphi)$ then also $t \in T^{\varphi,M}_{\text{dyn}}$ by definition of $T^{\varphi,M}_{\text{dyn}}$. Alternatively, by **(D2)**, we know $\text{places}_{\Downarrow}(\varphi) \subseteq D^{\varphi,M}_{\Downarrow}$, so if $t \in \text{places}_{\Downarrow}(\varphi)^-$ then again $t \in T^{\varphi,M}_{\text{dyn}}$. Thus, $t = w \in (T^{\varphi,M}_{\text{dyn}})^*$ as required. Now, let $n > 1$. For the sake of contradiction, let us assume that there exists an i such that $t_i \notin T^{\varphi,M}_{\text{dyn}}$ and let us assume that i is the largest index with this property. Clearly, t_i does not have any effect on the support places(φ), otherwise we can use the same argument as when $n = 1$ and show that $t_i \in T^{\varphi,M}_{\text{dyn}}$. Now, since w is a shortest trace, it must be the case that t_i enables t_j for some $i < j \leq n$. That is, t_i adds tokens to ${}^\bullet t_j$ and/or removes tokens from ${}^\circ t_j$. Let us consider each case:

1. If t_i removes tokens from ${}^\circ t_j$, then there exists $p \in {}^-t_i \cap {}^\circ t_j \cap C^M_{\Downarrow}$. Since j is greater than i (the last index such that $t_i \notin T^{\varphi,M}_{\text{dyn}}$) we have that $t_j \in T^{\varphi,M}_{\text{dyn}}$ which implies that $t_j \in {}^+D^{\varphi,M}_{\Uparrow}$ and/or $t_j \in (D^{\varphi,M}_{\Downarrow})^-$ by definition of $T^{\varphi,M}_{\text{dyn}}$. Let us consider each sub-case:

(a) If $t_j \in {}^+D^{\varphi,M}_{\Uparrow}$ then there exists $p' \in D^{\varphi,M}_{\Uparrow}$ such that $t_j \in {}^+p'$. By **(D4)** we have that ${}^\circ t_j \cap C^M_{\Downarrow} \subseteq D^{\varphi,M}_{\Downarrow}$ which in turn implies $p \in D^{\varphi,M}_{\Downarrow}$. Since $t_i \in p^-$, then $t_i \in (D^{\varphi,M}_{\Downarrow})^-$ and thus $t_i \in T^{\varphi,M}_{\text{dyn}}$ creating a contradiction.

(b) Hence, it must be the case that $t_j \in (D^{\varphi,M}_{\Downarrow})^-$. Similarly, this implies that there exists $p' \in D^{\varphi,M}_{\Downarrow}$ such that $t_j \in p'^-$. Now by **(D6)** we have that ${}^\circ t_j \cap C^M_{\Downarrow} \subseteq D^{\varphi,M}_{\Downarrow}$ which in turn implies $p \in D^{\varphi,M}_{\Downarrow}$. Again, since $t_i \in p^-$, then $t_i \in (D^{\varphi,M}_{\Downarrow})^-$ and thus $t_i \in T^{\varphi,M}_{\text{dyn}}$ creating a contradiction.

2. Hence, it must be the case that t_i adds tokens to ${}^\bullet t_j$, that is, there exists $p \in t_i^+ \cap {}^\bullet t_j \cap C^M_{\Uparrow}$. Again, since j is greater than i we have that $t_j \in T^{\varphi,M}_{\text{dyn}}$ which implies that $t_j \in {}^+D^{\varphi,M}_{\Uparrow}$ and/or $t_j \in (D^{\varphi,M}_{\Downarrow})^-$:

(a) If $t_j \in {}^+D^{\varphi,M}_{\Uparrow}$, then there exists $p' \in D^{\varphi,M}_{\Uparrow}$ such that $t_j \in {}^+p'$. So by **(D3)** we have that ${}^\bullet t_j \cap C^M_{\Uparrow} \subseteq D^{\varphi,M}_{\Uparrow}$ which in turn implies $p \in D^{\varphi,M}_{\Uparrow}$. Since $t_i \in {}^+p$, then $t_i \in {}^+D^{\varphi,M}_{\Uparrow}$ and thus $t_i \in T^{\varphi,M}_{\text{dyn}}$ creating a contradiction.

(b) Hence, it must be the case that $t_j \in (D^{\varphi,M}_{\Downarrow})^-$. This implies that there exists $p' \in D^{\varphi,M}_{\Downarrow}$ such that $t_j \in p'^-$. Now by **(D5)** we have that ${}^\bullet t_j \cap C^M_{\Uparrow} \setminus \{p'\} \subseteq D^{\varphi,M}_{\Uparrow}$, so unless $p = p'$, it must be the case that $p \in D^{\varphi,M}_{\Uparrow}$. Since $t_i \in {}^+p$, then $t_i \in {}^+D^{\varphi,M}_{\Uparrow}$ and thus $t_i \in T^{\varphi,M}_{\text{dyn}}$ creating a contradiction. Now finally, if $p = p'$ for all $p' \in D^{\varphi,M}_{\Downarrow}$ where $t_j \in p'^-$, then we shall use following facts:
   - $p \notin {}^\circ t_j$ (follows from case 1.)
   - $p \notin D^{\varphi,M}_{\Uparrow}$ (follows from case 2.a)
   - $W(p, t_j) = 1$, otherwise **(D7)** would imply $p \in D^{\varphi,M}_{\Uparrow}$
   - $p \notin \text{places}_{\Uparrow}(\varphi)$, otherwise $p \in D^{\varphi,M}_{\Uparrow}$ by **(D1)**

   We can thus conclude that $p \in \text{places}_{\Downarrow}(\varphi)$. This follows from the fact that $D^{\varphi,M}_{\Downarrow}$ is a least fixed point and the only remaining possible way for $p \in D^{\varphi,M}_{\Downarrow}$ is **(D2)**. Since none of the other cases applied, we can also conclude that t_j only appears in w in order to remove tokens from p. Moreover, t_i only appears in w in order to enable t_j by adding tokens to p. However, since $W(p, t_j) = 1$, this is always redundant for the purpose of satisfying φ and contradicts that w is some shortest trace.

As none of the cases above is possible, our original assumption that $t_i \notin T^{\varphi,M}_{\text{dyn}}$ is wrong, which implies that $w \in (T^{\varphi,M}_{\text{dyn}})^*$. □

Finally, let

$$P^{\varphi,M}_{\text{dyn}} = \text{places}(\varphi) \cup {}^\bullet T^{\varphi,M}_{\text{dyn}} \cup {}^\circ T \tag{2}$$

be a refined set of places with visible tokens. We remark that T is the set of all transitions and every inhibiting place must be preserved to uphold the invariant properties of the C-sets. We now consider the following abstraction.

Definition 7 (Dynamic Token Elimination). *The* dynamic token elimination *abstraction* $\alpha_{\text{dyn}}^{\varphi,M} : \mathcal{M}(N) \to \mathcal{M}(N)$ *is given by*

$$\alpha_{\text{dyn}}^{\varphi,M}(M)(p) = \begin{cases} M(p) & \text{if } p \in P_{\text{dyn}}^{\varphi,M} \\ 0 & \text{otherwise .} \end{cases}$$

Theorem 2 ($\alpha_{\text{dyn}}^{\varphi,M}$ Preserves Reachability). *Let M be a marking and* $\text{EF}\varphi$ *a reachability property. We have $M \vDash \text{EF}\varphi$ iff $\alpha_{\text{dyn}}^{\varphi,M}(M) \vDash \text{EF}\varphi$.*

In the example, Figure 1, we show all intermediary sets computed for the dynamic method on the depicted PN. In this example, the dynamic method will remove the tokens of p_1 while static method will not. The difference arises from the static method incorrectly assuming that t_1 is fireable in a future marking.

The following corollary is essential for the applicability of our technique in on-the-fly model-checking algorithms as it implies that our technique can be applied to any marking reached during traditional state-space exploration. The corollary follows directly from Lemma 3 and Theorem 2.

Corollary 1. *Given marking M_0 and reachability property* $\text{EF}\varphi$*, then $M_0 \vDash \text{EF}\varphi$ if and only if there exist markings $M_1, \ldots, M_n$ such that $M_n \vDash \varphi$ and $\alpha_{\text{dyn}}^{\varphi,M_{i-1}}(M_{i-1}) \xrightarrow{t_{i-1}} M_i$ and $t_{i-1} \in T_{\text{dyn}}^{\varphi,M_{i-1}}$ for all $1 \leq i \leq n$.*

The same corollary holds for the static abstraction $\alpha_{\text{sta}}^{\varphi}$, since the only difference is the sets $C_{\Uparrow}^{M}$, $C_{\Downarrow}^{M}$, and C_{F}^{M} restricting visibility propagation in the defining constraints.

4 Experimental Evaluation

We implemented our methods in the untimed engine of TAPAAL[2] [11] called `verifypn`[3] [2,7,10]. The methods are used in both the reachability and CTL algorithm. The reachability algorithm is a simple state-space exploration with heuristics [15] and here our method is applied to all successor states right before they are inserted into the frontier. The CTL algorithm is a top-down on-the-fly algorithm called CERTAINZERO [10]. The algorithm recursively breaks problems down into subproblems and structures them in a dependency graph [12,21] where each node consists of a marking and a sub-property. An example dependency graph is seen in Figure 2. The graph is explored in an on-the-fly manner until a conclusion for the root node is found. Starting from the initial assignment of false to all nodes, whenever all target nodes of a hyperedge have the value true, the source of the edge is assigned true. This continues until the root is assigned true or the process stabilizes at minimum fixed-point assignment on a fully-explore dependency graph. Note that a hyperedge with an empty set of

[2] TAPAAL is available at www.tapaal.net.
[3] `verifypn` is open-source on GitHub: https://github.com/TAPAAL/verifypn.

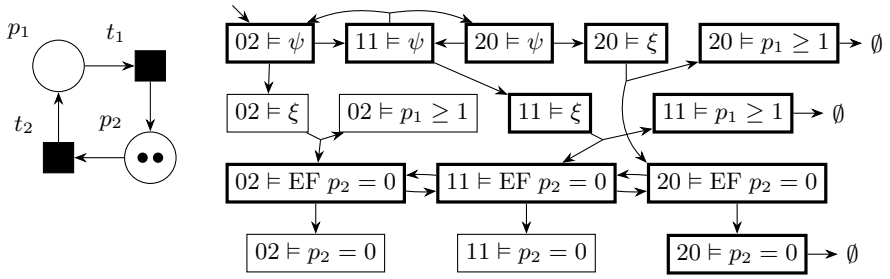


Fig. 2: Left: A Petri net with initial marking 02 (0 tokens in p_1, 2 tokens in p_2). Right: The full dependency graph generated by `verifypn` for the CTL property $\psi \equiv \mathrm{AF}\,\xi$ where $\xi \equiv (p_1 \geq 1 \wedge \mathrm{EF}\ p_2 = 0)$. The nodes that are assigned true in the minimum fixed-point assignment are drawn with thick borders.

target nodes always propagates the value true to its source. During the on-the-fly fixed-point computation, whenever a node containing a reachability property is generated, we apply the token elimination method to the associated marking. In Figure 2, the method would be applied to each node with the sub-property $\mathrm{EF}\,p_2 = 0$. Our techniques provide no benefits in this tiny example, but in other dependency graphs several reachability nodes may be simplified by our token elimination into the same node. Additionally, the simplified node likely has fewer dependencies due to the reduced number of successors, resulting in a smaller dependency graph.

Our tests use the Model Checking Contest (MCC) 2023 benchmark [18] that contains 131 different models. Most models are parameterized, amounting to 1 426 different instances of Petri nets[4]. For each instance, the benchmark provides 32 CTL properties and 32 reachability properties. This gives us 45 632 queries per category where 19 698 of the CTL queries contain a reachability subproperty.

We compare our implementations against the most recent version of Tapaal (`verifypn` without our methods) since Tapaal has consistently dominated the CTL category in the recent editions of the MCC competition [18]. We also briefly compare with the latest version of ITS-Tools [25] which got second place in MCC 2023. Each query is run with a timeout of 30 minutes and a memory limit of 15 GB on an AMD EPYC 7642 CPU. All experiments use the default settings of Tapaal, except for the search strategy which is set to random depth-first search (RDFS). We refer to the three versions of `verifypn` as different methods with Tapaal being the baseline. A reproducibility package containing binaries, data, and plotting scripts is available on Zenodo [16].

4.1 Results

As expected, our methods achieve multiple improvements in the CTL category, but marginal improvements in the reachability category of MCC. This is ex-

[4] The MCC2023 benchmark does not contain any Petri net with inhibitor arcs.

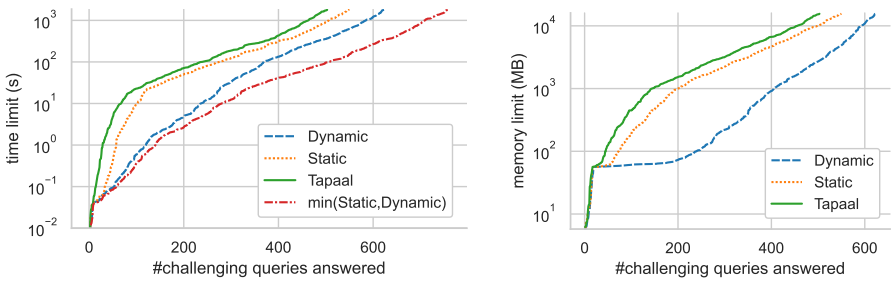
	Tapaal	Dynamic	Static	ITS-Tools
CTL (all)	32272	32392 (+120)	32319 (+47)	22998 (−9274)
CTLCardinality	16831	16923 (+92)	16849 (+18)	12056 (−4775)
CTLFireability	15441	15469 (+28)	15470 (+29)	10942 (−4499)

Table 1: Answers found by each method, per category. The numbers in parentheses show the difference to Tapaal.

pected since our methods are similar to the existing stubborn set reductions and structural reduction rules I and M in Tapaal [7], and those (together with all other optimization methods) are enabled. Similarly to our methods, structural reduction rules I and M maintain reachability properties. However, these two reductions alter the net's structure. Therefore, these rules are unsuitable for verifying reachability sub-properties of CTL properties unless we accept the overhead of allocating one or more new Petri nets. Our approach, on the other hand, only changes the immediate marking by removing tokens. This makes it applicable to reachability sub-properties of a CTL property without requiring extra allocations. Our CTL results are hence more interesting to examine and will be the main topic of discussion for the rest of this section.

The number of CTL answers found by each method and ITS-Tools is shown in Table 1. The dynamic method finds 120 more answers than Tapaal, whereas the static finds 47 more. When we distinguish between the cardinality and fireability subcategories, we see that the dynamic method finds more cardinality answers than the static one, but their additional answers with respect to Tapaal are similar for fireability. ITS-Tools finds 9274 fewer answers than Tapaal overall. As a result, we consider a comparison with ITS-Tools to be uninformative and we will exclude it in the following. The number of new answered queries may seem small compared to the size of the benchmark. However, a large portion of the queries in the MCC benchmark are straightforward to solve since the queries are randomly generated [18] and the state spaces of some models are small. For example, Tapaal solves 8275 queries using just query rewriting [2] and linear equations [13,22], and 29608 queries are solved in less than 30 seconds. However, some queries are too complex for any tool due to the extreme size of the state space [6]. For this reason, we shall from now on focus on queries where at least one of the three methods spends at least 2 minutes or 1 GB of memory. We classify such queries as *challenging*. There are 13,863 challenging queries in total, and 782 are answered by at least one of the three methods. A similar projection to challenging queries was made in [1], but our projection is not biased towards queries where our methods provide improvements, as it also includes queries where our method causes slowdowns.

The cactus plots in Figure 3 depict the time and memory usage on the challenging queries. In both plots, the static method curve resembles that of Tapaal but is shifted to the right, indicating that static often performs comparatively to Tapaal, but for some cases, it achieves major savings. The dynamic method



(a) Time (b) Memory

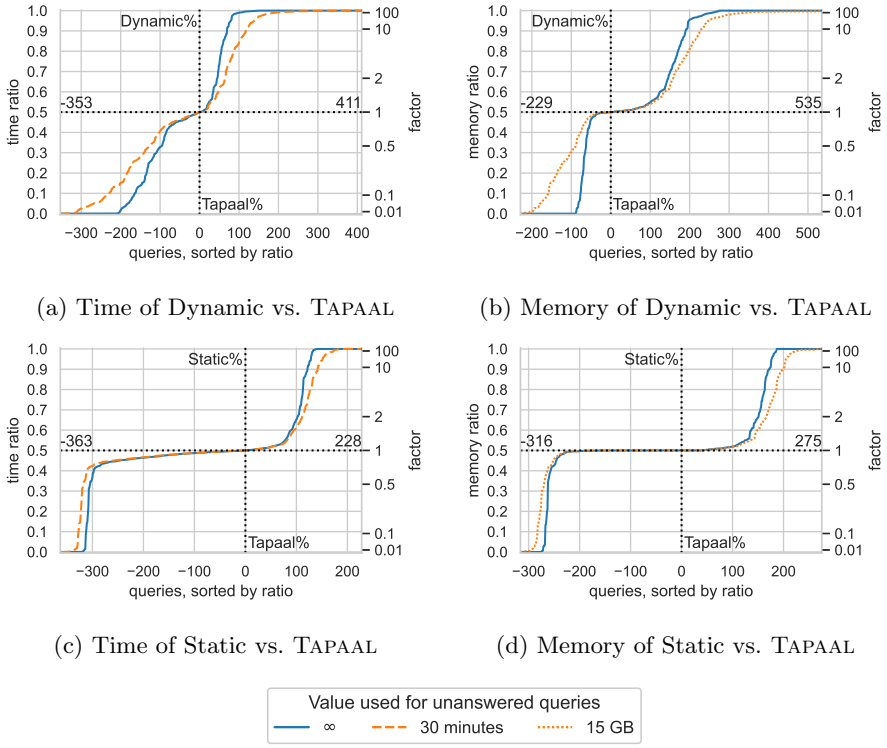
Fig. 3: Cactus plots showing the number of challenging queries that can be answered within the depicted time/memory limit. The plots are created by ordering the queries independently of the method (on x-axis) by the verification time or peak memory (on y-axis).

provides notable reductions in time and, especially, memory usage. The dynamic method solves more than 200 challenging queries within a 10-second limit, indicating that many queries that challenge TAPAAL become significantly easier with the dynamic method. Regarding memory, TAPAAL answers 144 challenging queries using less than 1 GB, while our dynamic method answers 405.

The dynamic and static methods answer different queries due to their different performance characteristics. For this reason, we also depict the minimum time of the static and dynamic method in Figure 3a which for each query considers the fastest time of our two methods. This is thus an approximation of running both methods in parallel. As shown, this version outperforms all the others by about 120 answers. The results indicate that the static and dynamic methods complement each other despite their similar definitions, and it may be useful to investigate strategies for selecting when to apply which in future work.

The number of configurations in the dependency graph explored by the CTL algorithm of `verifypn` has been significantly reduced by the dynamic method. Across all queries answered by all three methods, TAPAAL explores 880K configurations on average, the static method explores 866K (−1.7%), and dynamic method explores 807K (−8.3%). On the challenging queries answered by all three methods, TAPAAL explores 16.7M configurations on average, static explores 15.3M (−7.8%), and dynamic explores 10.1M (−39.3%).

Next, we examine the impact of our methods on a query-by-query basis using the challenging queries. To compare queries where one method finds an answer while the other does not, we consider two scenarios: (i) a pessimistic one where we assume that the unanswered queries require infinite time and memory, and (ii) an optimistic one where we assume that the unanswered queries use exactly the time/memory limit of our experiments. Both scenarios are shown on the ratio plots in Figure 4, where a value of 0.5 means that the methods used the same amount of time/memory on the same query. A value of 0.75 means that our



(a) Time of Dynamic vs. TAPAAL

(b) Memory of Dynamic vs. TAPAAL

(c) Time of Static vs. TAPAAL

(d) Memory of Static vs. TAPAAL

Fig. 4: Query-to-query comparison of time and memory usage on the challenging queries, showing the ratio between the TAPAAL baseline and one of our methods, i.e. $\frac{\text{TAPAAL}}{\text{TAPAAL+Ours}}$. Queries are sorted by the ratio and then offset such that 0 on the x-axis is where the methods perform equally. Hence, queries on the left (negative) are queries where TAPAAL performs the best and queries on the right (positive) are where our method performs the best. Each plot includes a pessimistic scenario (solid) where the unanswered queries require infinite resources and an optimistic (dashed or dotted) where unanswered queries require the resource limit of our tests. The axis on the right-hand side shows the ratio as a factor. I.e. a factor of 2 in subfigure (a) means our method was twice as fast.

method only used 25% of the summed time/memory while TAPAAL used 75%. We find that the performance impact of the dynamic method is generally more extreme in magnitude. It provides speedups for about half the queries. The dynamic method is 10 times faster (ratio of ≥ 0.9 on the graph) at 290 queries in the optimistic scenario and 341 queries in the pessimistic scenario, however, it is also 10 times slower than TAPAAL (ratio of ≤ 0.1) at 115 queries in the optimistic scenario and at 186 queries in the pessimistic scenario. When it comes to memory comparison in Figure 4b, the dynamic method provides many significant savings, matching what was concluded earlier. The dynamic method

Speed-up factor	All models ($N = 1426$)			Challenging models ($N = 473$)		
	Dynamic	Static	min(Dyn,Stat)	Dynamic	Static	min(Dyn,Stat)
≤ 0.01	122	45	23	41	17	8
≤ 0.1	224	59	30	64	18	8
≤ 0.5	544	91	57	139	22	13
≥ 2	449	313	488	106	69	119
≥ 10	271	115	287	83	38	86
≥ 100	222	85	236	78	34	81

Table 2: Number of models where at least one of the associated queries saw the given speedup or slowdown w.r.t. the TAPAAL baseline.

occasionally finds answers with degraded memory efficiency. Most degradations are likely attributed to the randomness of the search strategy, as the dynamic method does not accumulate memory usage. Alternatively, the query may have timed out, leading to inflated memory usage. The static method provides a time and memory efficiency almost identical to that of TAPAAL for a majority of the challenging queries. On Figure 4c, if we focus on queries where the methods have a significant performance difference, i.e. a ratio ≥ 0.6 or ≤ 0.4, equating to a 3:2=150% time usage by one of the methods, then the numbers favor the static method. The static method has 131 queries above 0.6 while TAPAAL has 82 queries below 0.4. This implies that the static method has some overhead, but it does not significantly impair its overall benefits. The memory plot in Figure 4d shows a small advantage of the static method over TAPAAL with about half of the queries having very similar peak memory usage, and of the remaining queries a larger portion sees significant memory savings with the static method (shown on the right of the plot).

Our method relies on the structure of the Petri nets, so we investigate whether the performance improvements are concentrated on a certain subset of the models. First, we extend the notion of *challenging* from queries to models. A model is *challenging* if at least 1/4th of its queries are challenging. In Table 2, we group queries by model and check whether at least one query in the group exceeds a speedup or slowdown factor of 2, 10, and 100. Out of the 1426 models, 222 (15.6%) feature a query solved two orders of magnitude faster by the dynamic method compared to TAPAAL, and similar speedups occur on 85 models (6.0%) when using the static method. The static method has significantly fewer queries that see slowdowns, especially small slowdowns to ≤ 0.5 speed (91 compared to the dynamic method's 544). Slowdowns occur on 122 (8.6%) of all models when using the dynamic method, about half as frequent as the two orders of magnitude speedups. On the 329 challenging models, we have 78 models (23.7%) that find a two-order-of-magnitude speedup for at least one query when using the dynamic method. These 78 models belong to 34 different families (a group of similar models scaled to various sizes) out of 131 in total. Again, we also consider the minimum of the dynamic and static methods as if they were run in parallel,

and we see that the dynamic and static method complement each other. The number of models with slowdowns is significantly reduced—only half as many as the static method alone which is otherwise the method with fewest slowdowns. Meanwhile, there are also a few additional models with high speedups than when considered individually.

Overall, it is clear that our methods amplify the verification of CTL properties through their reductions of the state space, often leading to time and, especially, memory savings compared to the state-of-the-art methods implemented in Tapaal. Since they only modify the immediate markings, we can use them in situations where other methods are not applicable.

5 Conclusion

We presented a new token elimination technique to reduce the state-space size in Petri net model checking. Our technique has two phases. First, we approximate sets of impossible transition firings and effects and then we compute sets of visible effects based on the possible firings and effects. This allows us to identify a set of transitions, such that the shortest trace to any goal marking consists exclusively of transitions from this set. Based on this, we derive a set of places where all tokens can be removed without affecting the satisfaction of a given reachability property. We also presented a static instance of our technique that, in contrast to the dynamic one, is computationally cheaper as it does not require recomputations based on the current marking. Since our technique only modifies the marking it can be used in contexts where other reductions are unsuitable while providing similar benefits.

We implemented our techniques in the state-of-the-art CTL algorithm of Tapaal, winner of the recent editions of the Model Checking Contest in the CTL category. Our implementation eliminates tokens in every reachability node according to the computed visible effects. Through experiments on the Model Checking Contest 2023 benchmark, we showed that our technique improves the performance of CTL model checking. The static technique improves time and memory usage, while the dynamic technique provides even greater time and memory savings at the cost of occasionally high overhead. Due to their different performance characteristics, running them in parallel is also a viable option, avoiding some of the time overhead.

While our experiments focus on the CTL properties, our technique can be adapted to other kinds of properties such as UpperBound queries which ask for the count of a maximum number of tokens in a given set of places. Not all tokens may be relevant to determine such upper bounds when using a state space search. Similarly, it can be used for liveness checks, as this can be rewritten to the CTL query asking if AG EF $enabled(t)$ for all t. Further applications of the token elimination technique to other types of logics are part of the future work.

References

1. Amat, N., Dal Zilio, S., Le Botlan, D.: Project and Conquer: Fast Quantifier Elimination for Checking Petri Net Reachability. In: Verification, Model Checking, and Abstract Interpretation: 25th International Conference, VMCAI 2024, London, United Kingdom, January 15–16, 2024, Proceedings, Part I. LNCS, vol. 14499, p. 101–123. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-50524-9_5
2. Bønneland, F., Dyhr, J., Jensen, P.G., Johannsen, M., Srba, J.: Simplification of CTL Formulae for Efficient Model Checking of Petri Nets. In: Khomenko, V., Roux, O.H. (eds.) Application and Theory of Petri Nets and Concurrency. LNCS, vol. 10877, pp. 143–163. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-91268-4_8
3. Bønneland, F.M., Jensen, P.G., Larsen, K.G., Muñiz, M., Srba, J.: Stubborn Set Reduction for Timed Reachability and Safety Games. In: Dima, C., Shirmohammadi, M. (eds.) Formal Modeling and Analysis of Timed Systems. LNCS, vol. 12860, pp. 32–49. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-85037-1_3
4. Bønneland, F.M., Jensen, P.G., Larsen, K.G., Muñiz, M., Srba, J.: Start Pruning When Time Gets Urgent: Partial Order Reduction for Timed Systems. In: Chockler, H., Weissenbacher, G. (eds.) Computer Aided Verification. LNCS, vol. 10981, pp. 527–546. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_28
5. Bønneland, F., Jensen, P., Larsen, K., Muñiz, M., Srba, J.: Stubborn Set Reduction for Two-Player Reachability Games. In: Logical Methods in Computer Science. vol. 17. International Federation for Computational Logic (Mar 2021). https://doi.org/10.23638/LMCS-17(1:21)2021
6. Burkart, O., Esparza, J.: More Infinite Results. In: Electronic Notes in Theoretical Computer Science. vol. 5, p. 29 (1997). https://doi.org/10.1016/S1571-0661(05)80680-2
7. Bønneland, F.M., Dyhr, J., Jensen, P.G., Johannsen, M., Srba, J.: Stubborn Versus Structural Reductions for Petri Nets. In: Journal of Logical and Algebraic Methods in Programming. vol. 102, pp. 46–63 (2019). https://doi.org/10.1016/j.jlamp.2018.09.002
8. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R., et al.: Handbook of Model Checking, vol. 10. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8
9. Czerwiński, W., Orlikowski, L.: Reachability in Vector Addition Systems is Ackermann-complete. In: Foundations of Computer Science. vol. 62, pp. 1229–1240. IEEE (2022). https://doi.org/10.1109/FOCS52979.2021.00120
10. Dalsgaard, A.E., Enevoldsen, S., Fogh, P., Jensen, L.S., Jepsen, T.S., Kaufmann, I., Larsen, K.G., Nielsen, S.M., Olesen, M.C., Pastva, S., Srba, J.: Extended Dependency Graphs and Efficient Distributed Fixed-Point Computation. In: van der Aalst, W., Best, E. (eds.) Application and Theory of Petri Nets and Concurrency. LNCS, vol. 10258, pp. 139–158. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-57861-3_10
11. David, A., Jacobsen, L., Jacobsen, M., Jørgensen, K.Y., Møller, M.H., Srba, J.: TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets. In: Flanagan, C., König, B. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. LNCS, vol. 7214, pp. 492–497. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28756-5_36

12. Enevoldsen, S., Larsen, K., Srba, J.: Abstract Dependency Graphs and Their Application to Model Checking. In: Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19). LNCS, vol. 11427, pp. 316–333. Springer-Verlag (2019). https://doi.org/10.1007/978-3-030-17462-0_18
13. Esparza, J., Melzer, S.: Verification of Safety Properties Using Integer Programming: Beyond the State Equation. vol. 16, pp. 159–189. Kluwer Academic Publishers (2000). https://doi.org/10.1023/A:1008743212620
14. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems, LNCS, vol. 1032. Springer Berlin, Heidelberg, 1 edn. (1996). https://doi.org/10.1007/3-540-60761-7
15. Jensen, J.F., Nielsen, T., Oestergaard, L.K., Srba, J.: TAPAAL and Reachability Analysis of P/T Nets, LNCS, vol. 9930, pp. 307–318. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53401-4_16
16. Jensen, N.Ø., Guldstrand Larsen, K., Srba, J.: Token Elimination in Model Checking of Petri Nets Reproducibility Package (Jan 2025). https://doi.org/10.5281/zenodo.14608439
17. Katz, S., Peled, D.: An Efficient Verification Method for Parallel and Distributed Programs. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. LNCS, vol. 354, pp. 489–507. Springer Berlin Heidelberg, Berlin, Heidelberg (1989). https://doi.org/10.1007/BFb0013032
18. Kordon, F., Bouvier, P., Garavel, H., Hulin-Hubard, F., Amat., N., Amparore, E., Berthomieu, B., Donatelli, D., Dal Zilio, S., Jensen, P., Jezequel, L., Paviot-Adet, E., Srba, J., Thierry-Mieg, Y.: Complete Results for the 2023 Edition of the Model Checking Contest. https://mcc.lip6.fr/2023/results.php (April 2023)
19. Kristensen, L.M., Schmidt, K., Valmari, A.: Question-Guided Stubborn Set Methods for State Properties. In: Formal Methods in System Design. vol. 29, pp. 215–251. Springer Science+Business Media (2006). https://doi.org/10.1007/s10703-006-0006-1
20. Liebke, T., Wolf, K.: Taking Some Burden Off an Explicit CTL Model Checker. In: Donatelli, S., Haar, S. (eds.) Application and Theory of Petri Nets and Concurrency. LNCS, vol. 11522, pp. 321–341. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-21571-2_18
21. Liu, X., Smolka, S.A.: Simple Linear-Time Algorithms for Minimal Fixed Points. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) Automata, Languages and Programming. LNCS, vol. 1443, pp. 53–66. Springer Berlin Heidelberg, Berlin, Heidelberg (1998). https://doi.org/10.1007/BFb0055040
22. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
23. Pastor, E., Cortadella, J., Roig, O.: Symbolic Analysis of Bounded Petri Nets. IEEE Transactions on Computers **50**(5), 432–448 (2001). https://doi.org/10.1109/12.926158
24. Peled, D.: All From One, One for All: On Model Checking using Representatives. In: Courcoubetis, C. (ed.) Computer Aided Verification. LNCS, vol. 697, pp. 409–423. Springer Berlin Heidelberg, Berlin, Heidelberg (1993). https://doi.org/10.1007/3-540-56922-7_34
25. Thierry-Mieg, Y.: Symbolic Model-Checking Using ITS-Tools. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. LNCS, vol. 9035, pp. 231–237. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_20

26. Thierry-Mieg, Y.: Structural Reductions Revisited. In: Janicki, R., Sidorova, N., Chatain, T. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 303–323. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-51831-8_15
27. Valmari, A.: A Stubborn Attack on State Explosion. In: Formal Methods in System Design. vol. 1, pp. 297–322. Kluwer Academic Publishers (1992). https://doi.org/10.1007/BF00709154

A Proofs for Section 2 (Petri Nets and the Reachability Problem)

Lemma 1. *Let $N = \langle P, T, W, I\rangle$ be a PN, $M, M' \in \mathcal{M}(N)$ markings, φ a state property, and $t \notin {}^{+}\text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)^{-}$ a transition. If $M \not\Vdash \varphi$ and $M \xrightarrow{t} M'$, then $M' \not\Vdash \varphi$.*

Proof. Let $N = \langle P, T, W, I\rangle$ be a PN, $M \in \mathcal{M}(N)$ a marking and φ a state property. For the sake of contradiction, assume that $M \not\Vdash \varphi$ and $M \xrightarrow{t} M'$ such that $M' \vDash \varphi$ for some $t \notin {}^{+}\text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)^{-}$. The proof proceeds by structural induction on φ.

- Case $\varphi = k \leq \sum_p v_p p$: Since $M \not\Vdash k \leq \sum_p v_p p$ then we know $k > \sum_p v_p M(p)$. In order for $M' \vDash \varphi$ it must be the case that $\sum_p v_p M'(p) > \sum_p v_p M(p)$. This is possible in two ways:
  - There exists place p' s.t. $v_{p'} > 0$ and $M'(p') > M(p')$. This implies that $t \in {}^{+}p'$. However, by definition $\text{places}_{\Uparrow}(\varphi) = \{p \in P \mid v_p > 0\}$ and thus $t \in {}^{+}\text{places}_{\Uparrow}(\varphi)$ creating a contradiction.
  - There exists place p' s.t. $v_{p'} < 0$ and $M'(p') < M(p')$. This implies that $t \in p'^{-}$. However, by definition $\text{places}_{\Downarrow}(\varphi) = \{p \in P \mid v_p < 0\}$ and thus $t \in \text{places}_{\Downarrow}(\varphi)^{-}$ creating a contradiction.

  Hence there is no $t \notin {}^{+}\text{places}_{\Uparrow}(\varphi) \cup \text{places}_{\Downarrow}(\varphi)^{-}$ such that $\sum_p v_p M'(p) > \sum_p v_p M(p)$, and so it must be that $M' \not\Vdash \varphi$.
- Case $\varphi = \varphi_1 \wedge \varphi_2$: Since $M \not\Vdash \varphi$ there exists $i \in \{1, 2\}$ s.t. $M \not\Vdash \varphi_i$. By definition ${}^{+}\text{places}_{\Uparrow}(\varphi_i) \subseteq {}^{+}\text{places}_{\Uparrow}(\varphi)$ and $\text{places}_{\Downarrow}(\varphi_i)^{-} \subseteq \text{places}_{\Downarrow}(\varphi)^{-}$ which by induction hypothesis implies $M' \not\Vdash \varphi_i$. By semantics of conjunction $M' \not\Vdash \varphi$ too, creating a contradiction.
- Case $\varphi = \varphi_1 \vee \varphi_2$: Since $M \not\Vdash \varphi$ then $M \not\Vdash \varphi_1$ and also $M \not\Vdash \varphi_2$. By definition ${}^{+}(\text{places}_{\Uparrow}(\varphi_1) \cup \text{places}_{\Uparrow}(\varphi_2)) \subseteq {}^{+}\text{places}_{\Uparrow}(\varphi)$ and $(\text{places}_{\Downarrow}(\varphi_1) \cup \text{places}_{\Downarrow}(\varphi_2))^{-} \subseteq \text{places}_{\Downarrow}(\varphi)^{-}$ which by induction hypothesis implies $M' \not\Vdash \varphi_1$ and $M' \not\Vdash \varphi_2$. By semantics of disjunction $M' \not\Vdash \varphi$ too, creating a contradiction. □

B Proofs for Section 3 (Token Elimination)

Theorem 1 ($\alpha^{\varphi}_{\text{sta}}$ Preserves Reachability). *Let M be a marking and $\text{EF}\varphi$ a reachability property. We have $M \vDash \text{EF}\varphi$ iff $\alpha^{\varphi}_{\text{sta}}(M) \vDash \text{EF}\varphi$.*

Proof. The proof of Theorem 1 is a special instance of Theorem 2 and we refer to the proof of that theorem.

Lemma 2. *Let $p \in P$ be a place and $t \in T$ be a transition.*

- *If $p \notin C^{M}_{\Uparrow}$ then $M'(p) \leq M(p)$ for all M' s.t. $M \to^{*} M'$,*
- *If $p \notin C^{M}_{\Downarrow}$ then $M'(p) \geq M(p)$ for all M' s.t. $M \to^{*} M'$,*
- *If $t \notin C^{M}_{F}$ then $M' \overset{t}{\not\to}$ for all M' s.t. $M \to^{*} M'$.*

Proof. We prove all three claims by showing that if $t \in T$ is a fireable transition, i.e. if $M \xrightarrow{wt}$ for some $w \in T^*$, then $t \in C_F^M$ and $t^+ \subseteq C_{\Uparrow}^M$ and ${}^-t \subseteq C_{\Downarrow}^M$. The proof is by induction on the length of w. t If $|w| = 0$, then t is enabled in M, i.e. $W(p, t_i) \leq M(p)$ and $I(p, t_i) > M(p)$ for all $p \in P$. Thus, by constraint **(C)**, we have $t \in C_F^M$ and $t^+ \subseteq C_{\Uparrow}^M$ and ${}^-t \subseteq C_{\Downarrow}^M$.

Now let $|w| > 0$. If t is enabled in M, then as by the base case the claim holds. Otherwise, we have that t is enabled after firing w. Hence, for all $p \in P$, either $W(p, t) \leq M(p)$ or there must exist a transition t_w in w such that $p \in t_w^+$. Similarly, for all $p \in P$, either $I(p, t) > M(p)$ or there must exists a t'_w in w such that $p \in {}^-t'_w$. By the induction hypothesis, we have $t'^+_w \subseteq C_{\Uparrow}^M$ and ${}^-t'_w \subseteq C_{\Downarrow}^M$. Now by constraint **(C)**, it follows that $t \in C_F^M$ and $t^+ \subseteq C_{\Uparrow}^M$ and ${}^-t \subseteq C_{\Downarrow}^M$. □

Theorem 2 ($\alpha_{\text{dyn}}^{\varphi,M}$ Preserves Reachability). *Let M be a marking and* $\text{EF}\varphi$ *a reachability property. We have $M \vDash \text{EF}\varphi$ iff $\alpha_{\text{dyn}}^{\varphi,M}(M) \vDash \text{EF}\varphi$.*

Proof. Let $\bar{M} = \alpha_{\text{dyn}}^{\varphi,M}(M)$.

$\Rightarrow$: We argue that $M \vDash \text{EF}\varphi$ implies that $\bar{M} \vDash \text{EF}\varphi$. Assume $M \vDash \text{EF}\varphi$, then Lemma 3 implies that there must exist $w \in (T_{\text{dyn}}^{\varphi,M})^*$ such that $M \xrightarrow{w} M'$ for some $M' \vDash \varphi$. The proof proceeds by showing that $\bar{M} \xrightarrow{w} \bar{M}'$ for some $\bar{M}' \vDash \varphi$. For the sake of contradiction, assume that $\bar{M} \not\xrightarrow{w}$. This means that there exists a transition t_w in w that was disabled by $\alpha_{\text{dyn}}^{\varphi,M}$. This can happen only in two ways:

- There exists a $p \in {}^\bullet t_w$ such that $M(p) > \bar{M}(p)$. But this is not possible since $t_w \in T_{\text{dyn}}^{\varphi,M}$ and ${}^\bullet T_{\text{dyn}}^{\varphi,M} \subseteq P_{\text{dyn}}^{\varphi,M}$, which implies that $M(p) = \bar{M}(p)$ for all $p \in {}^\bullet t_w$.
- There exists a $p \in t_w^\circ$ such that $M(p) < \bar{M}(p)$. But this is not possible since $T^\circ \subseteq P_{\text{dyn}}^{\varphi,M}$.

Hence, it must be the case that $\bar{M} \xrightarrow{w}$. Since $\text{places}(\varphi) \subseteq P_{\text{dyn}}^{\varphi,M}$, the tokens in $\text{places}(\varphi)$ are not modified by $\alpha_{\text{dyn}}^{\varphi,M}$, so combined with the effect of firing w we have that $M'(p) = \bar{M}'(p)$ for all $p \in \text{places}(\varphi)$ and therefore $\bar{M}' \vDash \varphi$ implying that $\bar{M} \vDash \text{EF}\varphi$.

$\Leftarrow$: We argue that $\bar{M} \vDash \text{EF}\varphi$ implies that $M \vDash \text{EF}\varphi$. By definition of $\alpha_{\text{dyn}}^{\varphi,M}$, it is clear that tokens are only removed, but never from T°. Hence, $\alpha_{\text{dyn}}^{\varphi,M}$ only removes behavior, and any trace fireable in $\bar{M}$ is also fireable in M. So assuming $\bar{M} \xrightarrow{w} \bar{M}'$ with $\bar{M}' \vDash \varphi$ then also $M \xrightarrow{w} M'$. And again, since $\text{places}(\varphi) \subseteq P_{\text{dyn}}^{\varphi,M}$, the tokens in $\text{places}(\varphi)$ are not modified by $\alpha_{\text{dyn}}^{\varphi,M}$, the effect of firing w results in $M'(p) = \bar{M}'(p)$ for all $p \in \text{places}(\varphi)$ and thus $M' \vDash \varphi$ implying that $M' \vDash \text{EF}\varphi$. This concludes the proof for Theorem 2. Theorem 1 can be proved in a similar manner with the relaxation that $C_{\Uparrow}^M = C_{\Downarrow}^M = P$ and $C_F^M = T$. □