Structural Reductions of Colored Petri Nets and P/T Nets with Inhibitor Arcs

Jesper A. van Diepen Nicolaj Ø. Jensen Mathias M. Sørensen

Cassiopeia, Department of Computer Science, Aalborg University, Denmark
{jvandi15, naje17, mmsa17}@student.aau.dk

Thesis Summary

System correctness of critical systems is crucial. One method to ensure that a system adheres to its specification is through the method of model checking, which is an algorithmic approach to verify properties of systems. However, most real-life systems are complicated with multiple concurrent components, which means that there is a huge number of states. This is known as the state space explosion problem, and one method to mitigate it is to perform structural reductions, where the structure of the system is analyzed and reduced while preserving the properties in question.

P/T nets and colored Petri nets are model formalisms for modeling concurrent systems, and in this thesis we present novel structural reduction rules for both, extensions to some existing P/T net reduction rules, and we generalize a handful of existing P/T net reductions to colored Petri nets. In regard to structural reductions on P/T nets, we extend Rule C from [5] by Bønneland et al. to be more general and therefore applicable in more nets. We also define a novel Rule S for P/T nets that combines the concept of a free agglomeration [28] and atomicity from Rule R in [10]. For colored petri nets we introduce rules C, D, E, F, I, Q, and U. Rules D, E, F, I, and Q are generalizations of corresponding rules from [5], such that the rules are applicable on colored Petri nets. Colored Rule C is a generalization to colored Petri nets on top of the extension mentioned earlier.

Rule U is a generalization of the atomic free agglomeration Rule S to colored Petri nets. However, we restrict Rule U to only handle equally weighted arcs, as we found this version to perform better.

We prove the correctness of all rules we present in this thesis for either the logic CTL* or one of its fragments. This is done using concepts like bisimilarity, stutter-bisimilarity, and stutter-equivalent traces. We also implement the rules in verifypn, which is the verification engine in the Petri net model checking tool TAPAAL. TAPAAL previously had no color reduction rules.

We do extensive testing of all rules, and several rule application sequences, where we apply the rules in different permutations. The testing is performed with the Model Checking Contest 2021 model suite. For the structural rules on P/T nets we use the 1181 P/T nets, and for the rules on colored petri nets, we use the 230 colored Petri nets. There are 6 categories of 16 queries available for each model about cardinality and fireability for CTL, LTL, and reachability logic. This gives us a total of 113.376 queries for our P/T net experiments and 22.080 queries for our colored Petri net experiments. During testing we measure metrics such as rule applicability, the time it takes to reduce the net, and time taken in verification of the property. For colored Petri net experiments we also measure the time taken to apply the colored structural reductions and the time it takes to unfold the colored Petri net. We compare the results we obtain from using our rules, to a base experiment that consists of using verifypn without our rules. We present and discuss results mainly from the reachability fireability category, but also summarize results from the other categories.

We check the correctness of our implementation of our rules, by making sure the answer to the queries are the same between experiments with and without our rules applied. Additionally, we test the correctness of all rules on a modified version of the MCC2021 models containing inhibitor arcs. We find that our rules are correct, also on models with inhibitor arcs.

Regarding performance, we find that Rule S on P/T nets get 393 more answers than the baseline. These answers are primarily from the queries in the reachability category. On CPNs we gain 162 more answers using the rule sequence IUC, which is the best performing permutation. We generally see that we are faster in unfolding. State space size is a lot better, which is what we expect.

The thesis starts with an introduction followed by preliminaries and definitions of CTL* and Petri nets. Then we present the concepts needed for structural reductions and the setup of our benchmark. The P/T net reduction rules and colored Petri net reductions rules are then presented in two separate sections, and each of these sections ends with a presentation of our results using the rules of that section. Finally, we discuss and conclude on our rules and findings and then propose some future work.

Structural Reductions of Colored Petri Nets and P/T Nets with Inhibitor Arcs

Jesper A. van Diepen Nicolaj Ø. Jensen Mathias M. Sørensen

Cassiopeia, Department of Computer Science, Aalborg University, Denmark
{jvandi15, naje17, mmsa17}@student.aau.dk

Abstract Structural reductions of Petri nets allow for more efficient model checking by mitigating the state space explosion problem. For a colored Petri net (CPN) there is an additional potential explosion in the size of the net during unfolding, and therefore structural reduction of CPNs is an effective technique to improve model checking performance.
We present novel structural reduction rules for both P/T nets and CPNs as well as generalizations of existing reduction rules. These include removal of parallel structures, removal of redundant structures, and agglomerations. We prove the correctness of all presented rules and extensions, and implement the rules in `verifypn`, the verification engine of TAPAAL. To test the implementation and performance of our rules we apply the rules on models from MCC2021. We measure multiple metrics during testing and compare these metrics against a baseline without our presented rules. We find that our Rule S, an atomic free agglomeration, provide 393 additional answers across all categories on the P/T nets. Our best rules on CPN models, Rule I, U, and C, provide an additional 162 answers across all categories. We conclude that many of our colored rules are generalizations of rules on P/T nets, and does the same reduction, but on a higher abstraction level. However, there is next to no overhead in applying the colored rules on CPNs, and we gain time during unfolding and reducing.

Keywords: *Colored Petri net, P/T net, structural reductions, MCC2021, agglomeration, LTL, CTL, Reachability, model checking, inhibitor arcs*

1 Introduction

Model checking [7] is a computer-aided method to verify properties in a model of one or more systems and processes, e.g. a transmission protocol, the activation of an airbag, or a traffic light. For a traffic light, important properties include that *all lanes eventually have green light* or that *two crossing lanes do not have a green light at the same time* among others. Model checking is especially useful in critical systems, where reliability and correctness are main priorities. The simplest form of model checking is searching through the state space of the model until we can verify that the model satisfies the given property. One of the main difficulties in model checking, therefore, lies in the size of the state space. Model checking, unfortunately, suffers from the *state space explosion* problem [7], especially when modelling concurrent processes, as all the possible interleavings of these introduce a large number of states, that has to be explored.

One way of reducing the state space explosion is by using structural reductions [7, 28, 5]. The idea in this approach is that we analyse the structure of the given model to remove redundant parts of the model with respect to the property that we wish to verify. Some structural reductions lead to an exponential reduction in the state space size, while others simply make the net smaller, saving us processing resources and memory usage.

P/T nets (or Petri nets) [25] are a modelling formalism that can be used to model concurrent systems, and many extensions exist incorporating time [15], colors/types [19], and games [9]. P/T nets have gained popularity due to their graphical nature and their intuitiveness for modelling concurrent processes. A petri net is a bipartite graph with two types of nodes: places and transitions. Arcs connect a place to a transition (input arc) or a transition to a place (output arc). Each place is marked with a non-negative number of tokens, and the marking across all places makes up the state of the net. When the transition is fired, the marking is updated by removing (consuming) tokens from the input places and adding (producing) tokens to the output places. To fire a transition, all places with arcs to the given transition must have one or more tokens. Arcs can also be weighted changing how many tokens a transitions consumes and produces in each place. An inhibitor arc is a special kind of arc from a place to a transition and it inhibits the firing of the given transition if the place contains tokens equal to or more than the weight on the inhibitor arc. Tokens thus elegantly model resources, signals, or messages while transitions models actions and synchronization.

In this thesis we also focus on colored Petri nets (CPN) with inhibitor arcs. In a CPN each token has a color, i.e. a type, and transitions can have guards restricting which types of tokens it consumes and produces. A CPN hereby allows additional abstractions resulting in more compact nets. Every CPN can be unfolded into an equivalent P/T net [20]. This is typically desirable as many model checking techniques are developed for P/T nets. However, this unfolding often results in much larger nets. It is therefore beneficial to apply structural reductions on the CPNs, as the unfolded net is potentially exponentially smaller, and the unfolding itself will be faster with a smaller CPN.

1.1 Our contributions

We present novel structural reductions and extensions of structural reductions for both colored Petri nets and P/T nets. We define and prove the correctness of two new structural reduction rules for P/T nets with inhibitor arcs. One of which is a relaxation of the Rule C from [5] by Bønneland et al. that removes places in parallel. The other is a novel atomic free agglomeration, Rule S, which is more general than a free agglomeration as defined by Thierry-Mieg in [28], since it is able to remove one consumer at a time and handle in- and out arcs with different weights. This is based on the similar atomic post agglomeration described in [10]. We then generalize these two rules as well as four rules by Bønneland et al. [5] and one rule by Thierry-Mieg [28] to colored Petri nets with inhibitor arcs. We prove the correctness of each rule. Table 1 shows our structural reduction rules and which categories of properties they preserve. We also implement the rules in the engine of the verification tool TAPAAL, which previously had no reduction rules for colored Petri nets. Finally, we also test the rules on the Petri nets of Model Checking Contest [21] and compare the new version of TAPAAL that includes our rules, to a baseline which does not use our new rules.

	Supported properties					
P/T rules	Reach	LTL	CTL	CTL*	Stutter	Deadlock
Rule C: Parallel P	✓	✓	✓	✓	✓	✓
Rule S: Atomic F agglo	✓	-	-	-	-	-
Rule T: Pre agglo	✓	✓	-	-	-	✓
CPN rules						
Rule C: Parallel P	✓	✓	✓	✓	✓	✓
Rule D: Parallel T	✓	✓	✓	✓	✓	✓
Rule E: Dead T	✓	✓	✓	✓	✓	✓
Rule F: Redundant P	✓	✓	✓	✓	✓	✓
Rule I: Irrelevant P+T	✓	-	-	-	-	-
Rule S: Atomic F agglo	✓	-	-	-	-	-
Rule T: Pre agglo	✓	✓	-	-	-	✓
Rule U: Atomic F agglo k=1	✓	-	-	-	-	-
Rule Q: Prefire T	✓	✓	✓	✓	-	✓

Table 1: Rules and the properties they preserve. The Stutter column refers to support for the neXt operator, and similarly for the Deadlock column refers to support for the deadlock proposition. E.g. Rule Q preserves CTL*\X. Rules which are not novel are given a gray background. They are still included since they were implemented by us and will be used for comparison.

1.2 Related Work

Berthelot in [2, 1] introduces some of the first structural reductions on Petri nets with a focus on preserving liveness and boundedness of the net. Later in [23], Murata presents a survey of structural reductions, related techniques such as linear equations and marked graphs, as well as a wider range of Petri net properties. These techniques are the foundation of the model checking tool TAPAAL and its structural reductions which is presented by Bønneland et al. in [5]. In [10] by van Diepen et al. these reductions are extended and additional reductions are added, some focusing on inhibitor arcs. Structural reductions can also be combined with SMT solvers and over- and under-approximations in a counterexample-guided abstraction refinement framework as shown by Thierry-Mieg in [28]. A special type of structural reductions, called agglomerations, are heavily studied by Haddad et al. [13, 14]. Agglomerations reduce interleavings in the underlying transition system and thus reduce the number of states significantly. Agglomerations in CPNs for transitions without guards have been presented by Haddad et al. in [13].

This work was extended to consider the syntax of guard- and arc expressions by Evangelista et al. in [12]. In this thesis we present an atomic free agglomeration for CPNs that considers guards, generalizing Thierry-Mieg's free agglomeration [28] to CPNs and making it atomic, a concept introduced by van Diepen et al. in [10].

1.3 Thesis outline

In Section 2 we give a definition of transitions systems as well as the syntax and semantics of CTL* and its fragments. Then in Section 3 we define colored Petri nets and their components, as well as the special case of a colored Petri net with just one color, called a P/T net. We give a brief overview of structural reductions and the setup of our experiments in Section 4. Then in Section 5, we introduce our work on structural reductions on P/T nets, and in the subsection, Section 5.3, we discuss our results using those rules. Similarly, we then define our novel colored structural reduction rules in Section 6, and discuss our results using those rules in Section 6.8. We discuss overall findings and results in Section 7 and conclude on our findings in Section 8. Lastly, we present our ideas for future work in Section 9. We only show a subset of the results and figures generated from testing during the discussion of the results. The full range of figures can be found in Appendix C, showing results on other logics, and tables not included in the main text.

Bibliographic Remarks Parts of this thesis originate from our previous work in [10]. In particular, the two opening paragraphs of the introduction of Section 1, major parts of Section 2.2 about the formalism CTL* and its fragments, the definition of labeled transition systems in Section 2.1, the definitions of $places(\varphi)$, invisibility, and correctness and related paragraphs in Section 4, as

well as the description of DEIS MCC in Section 4.2. Some sentences in the aforementioned parts have been rephrased in order to fit this thesis and typos have been fixed.

2 Preliminaries

Let $\mathbb{N}_0$ be all natural numbers including 0, and let $\mathbb{N}_\infty$ be the natural numbers including ∞.

2.1 Labeled transition systems

Definition 1 (Labeled Transition System). *A labeled transition system (LTS) is a triple $TS = \langle \mathcal{S}, A, \rightarrow \rangle$ where*

- *$\mathcal{S}$ is a non-emtpy set of states,*
- *A is a set of actions, and*
- *$\rightarrow \subseteq \mathcal{S} \times A \times \mathcal{S}$ is a transition relation.*

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 $s' \in S$ and say that a is *enabled* in s. We write $s \rightarrow s'$ if there exists an action $a \in A$ such that $s \xrightarrow{a} s'$, and $s \not\rightarrow$ if no such action a exists. For a state s where $s \not\rightarrow$ we say that it is a *dead* state or in a *deadlock*. We extend the relation $\xrightarrow{a}$ inductively to traces $w \in A^*$ such that $s \xrightarrow{\varepsilon} s$ and $s \xrightarrow{wa} s'$ if $s \xrightarrow{w} s''$ and $s'' \xrightarrow{a} s'$. We write $s \rightarrow^n s'$ if there exists a $w \in A^n$ such that $s \xrightarrow{w} s'$, and we write $s \rightarrow^* s'$ if $s \rightarrow^n s'$ for some $n \geq 0$. If $w \in A^\omega$ it is an infinite trace. Given a set X we let $X^\bigstar = X^* \cup X^\omega$ be all finite and infinite sequences of elements from X. We say that a is *fireable* if s_0 is the initial state of an LTS and $s_0 \rightarrow^* s' \xrightarrow{a}$.

2.2 CTL* and its fragments

In this section we introduce the formalism *computation tree logic star* (CTL*) [11], which is a superset of *computation tree logic* (CTL) [6] and *linear temporal logic* (LTL) [26]. We also cover these two logics, as well as *reachability* (Reach), and *stutter-sensitive properties*. The relationship between these logics can be seen in Figure 1.

2.2.1 CTL* There are two types of CTL* formulas: *state formulas* and *path formulas*. Formally, a CTL* formula has one of the following forms:

$$\varphi ::= \top \mid \pi \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid A\psi_1 \mid E\psi_1$$
$$\psi ::= \varphi_1 \mid \neg\psi_1 \mid \psi_1 \wedge \psi_2 \mid X\psi_1 \mid F\psi_1 \mid G\psi_1 \mid [\psi_1 U \psi_2]$$

where φ is the state formulas, ψ is the path formulas, and $\pi \in \Pi$ is an atomic proposition. In the context of P/T nets where states are markings, such a proposition π has the form $\alpha M \bowtie k$, where M is the given marking, $\alpha \in \mathbb{R}^P$

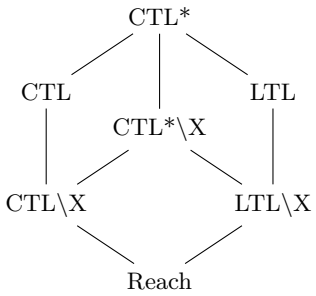


Figure 1: Logic formalisms and their relationship in a lattice.

is a vector of coefficients, and $k \in \mathbb{R}$ a constant, and $\bowtie \in \{<, \leq, =\geq, >\}$ is a comparison operator. Often we write αM as a linear equation using the place names to represent the number of tokens at that place, e.g. $2p_1 + p_2 \leq 5$. Two special propositions are $en(t)$, that asserts that the transition $t \in T$ is enabled, and *deadlock*, that asserts that no transition is enabled. Furthermore, A (all) and E (exists) are *path quantifiers*, and X (next), F (eventually), G (globally), U (until) are *temporal operators*. Additional boolean and temporal operators can be derived.

CTL* describes properties of LTSs. Let $TS = \langle \mathcal{S}, A, \rightarrow \rangle$ be an LTS with initial state s_0. In the semantics below, we let $\lambda \in \mathcal{S}^{\bigstar}$ denote a path such that $\lambda_0 \rightarrow \lambda_1 \rightarrow \ldots$ and let $\lambda[i]$ denotes the sub-path of λ starting from λ_i. A *maximal path* λ' is either infinite or ends in a deadlock. For state formulas the semantics are defined inductively such that $s \vDash \varphi$ denotes a state s satisfying the state formula φ.

$s \vDash \top \quad \equiv \top$
$s \vDash \pi \quad \equiv \pi(s)$
$s \vDash \neg\varphi \quad \equiv s \nvDash \varphi$
$s \vDash \varphi_1 \wedge \varphi_2 \equiv (s \vDash \varphi_1) \wedge (s \vDash \varphi_2)$
$s \vDash A\psi \quad \equiv \lambda \vDash \psi$ for all maximal paths λ starting in s
$s \vDash E\psi \quad \equiv \lambda \vDash \psi$ for some maximal path λ starting in s

Path formulas are also defined inductively, such that $\lambda \vDash \psi$ denotes path λ satisfying path formula ψ.

$\lambda \vDash \varphi \quad \equiv \lambda_1 \vDash \varphi$
$\lambda \vDash \neg\psi \quad \equiv \lambda \nvDash \psi$
$\lambda \vDash \psi_1 \wedge \psi_2 \equiv (\lambda \vDash \psi_1) \wedge (\lambda \vDash \psi_2)$
$\lambda \vDash X\psi \quad \equiv \lambda[1] \vDash \psi$
$\lambda \vDash F\psi \quad \equiv \lambda[n] \vDash \psi$ for some $n \geq 0$
$\lambda \vDash G\psi \quad \equiv \lambda[n] \vDash \psi$ for all $n \geq 0$
$\lambda \vDash \psi_1 U \psi_2 \equiv (\lambda[k] \vDash \psi_1) \wedge (\lambda[n] \vDash \psi_2)$ for some $n \geq 0$, for all $k, 0 \leq k < n$

We write $TS \vDash \varphi$ iff $s_0 \vDash \varphi$.

Given a set of atomic propositions Π and a labeling function $L : \mathcal{S} \rightarrow 2^{\Pi}$, we can label each state of a transition system with a subset of propositions, which

are true in the given state. This can be used to determine if two transition systems are bisimilar and satisfy the same CTL* properties.

Definition 2 (Bisimilarity [3]). *Given two LTSs* $TS = \langle \mathcal{S}, A, \rightarrow \rangle$ *and* $TS' = \langle \mathcal{S}', A', \rightarrow' \rangle$ *with initial states* s_0 *and* s'_0, TS *and* TS' *are said to be bisimilar w.r.t some set* Π *of atomic propositions and two labeling functions* $L : \mathcal{S} \rightarrow 2^{\Pi}$ *and* $L' : \mathcal{S}' \rightarrow 2^{\Pi}$, *if there exists a bisimulation* $\mathcal{R}$ *such that* $s_0 \mathcal{R} s'_0$. *A bisimulation* $\mathcal{R}$ *is a relation on* $\mathcal{S} \times \mathcal{S}'$ *such that whenever* $s_1 \mathcal{R} s'_1$ *then:*

1. $L(s_1) = L'(s'_1)$,
2. *if* $s_1 \rightarrow s_2$ *then there exists a* s'_2 *such that* $s'_1 \rightarrow s'_2$ *and* $s_2 \mathcal{R} s'_2$, *and*
3. *if* $s'_1 \rightarrow s'_2$ *then there exists a* s_2 *such that* $s_1 \rightarrow s_2$ *and* $s_2 \mathcal{R} s'_2$.

Theorem 1 (LTS Equivalence [3]). *Given two states* s, s', *and a bisimulation* $\mathcal{R}$ *such that* $s \mathcal{R} s'$, *then* $s \vDash \varphi$ *iff* $s' \vDash \varphi$ *for all* $\varphi \in CTL^*$.

2.2.2 CTL CTL is a fragment of CTL* that describes branching properties of systems. This is reflected in the syntax, specifically, all temporal operators must be preceded by a path quantifier. CTL formulas have the following form:

$$\varphi ::= \top \mid \pi \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid AX\varphi_1 \mid AF\varphi_1 \mid AG\varphi_1 \mid A[\varphi_1 U \varphi_2] \mid \\ EX\varphi_1 \mid EF\varphi_1 \mid EG\varphi_1 \mid E[\varphi_1 U \varphi_2]$$

The semantics of the operators are unchanged from description in the earlier section.

2.2.3 LTL LTL is another fragment of CTL*. This fragment of CTL* describes properties of traces and uses no path quantifiers, except for an implicitly prefixed A operator. LTL formulas have the following form:

$$\psi ::= \pi \mid \neg\psi_1 \mid \psi_1 \wedge \psi_2 \mid X\psi_1 \mid F\psi_1 \mid G\psi_1 \mid [\psi_1 U \psi_2]$$

There are two types of semantics for LTL. The first semantic is the one presented earlier for CTL*, and the alternative LTL semantics considers only infinite traces. Traces that normally end in a deadlock are extended by letting the system repeat the deadlock state infinitely, i.e. the system stutters infinitely. The Model Checking Competition [21] uses the alternative semantics. The structural reductions presented in this thesis work for both semantics.

2.2.4 Stuttering To proceed, we first define stuttering and stutter equivalence. Given a set of atomic propositions Π and a labeling function $L : \mathcal{S} \rightarrow 2^{\Pi}$, a *stutter* occurs when a transition system transitions from one state to another state where the same subset of the atomic propositions are true, i.e. if $s \rightarrow s'$ and $L(s) = L(s')$.

Properties that use the temporal next operator X are called stutter-sensitive properties. Not all reductions are applicable when checking stutter-sensitive properties and these reductions are therefore called stutter insensitive [24].

We use CTL*\X, CTL\X, and LTL\X to denote stutter-insensitive CTL*, CTL, and LTL without the next operator X. Clearly, these are respectively proper subsets of CTL*, CTL, and LTL.

Definition 3 (Stutter equivalence). *Let Π be a set of atomic propositions, and let $\sigma, \sigma' \in (2^{\Pi})^{\star}$ be two sequences of subsets of propositions such that $\sigma = \rho_0, \rho_1, \rho_2, \ldots$ and $\sigma' = \rho'_0, \rho'_1, \rho'_2, \ldots$. Then σ and σ' are said to be* stutter equivalent *iff there exists two infinite sequences $0 = i_0 < i_1 < \ldots$ and $0 = j_0 < j_1 < \ldots$ such that for all $k \geq 0$ we have $\rho_{i_k} = \rho_{i_k+1} = \cdots = \rho_{i_{k+1}-1} = \rho'_{j_k} = \rho'_{j_k+1} = \cdots = \rho'_{j_{k+1}-1}$*

Given an LTS $\langle \mathcal{S}, A, \rightarrow \rangle$, a set of atomic propositions Π, and a labeling function $L : \mathcal{S} \rightarrow 2^{\Pi}$, a path $\lambda \in \mathcal{S}^{\star}$ defines a sequence of subsets of propositions $L(\lambda_1), L(\lambda_2), \ldots$. Hence we say that two paths $\lambda, \lambda' \in \mathcal{S}^{\star}$ are stutter equivalent w.r.t. L if their underlying sequences of subsets of propositions are stutter equivalent. Similarly, a trace $w \in A^{\star}$ where $s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} s_2 \ldots$ for some initial state $s_0 \in \mathcal{S}$ defines a path visiting states $s_0, s_1, \ldots$. Hence, we say that two traces $w, w' \in A^{\star}$ are stutter equivalent w.r.t. L if the paths that they visit are stutter equivalent w.r.t. L.

Theorem 2 (LTL stutter invariance [26]). *Let $\sigma, \sigma' \in (2^{\Pi})^{\star}$ be two stutter equivalent sequences of subsets of propositions, and let ψ be an LTL\X formula. Then $\sigma \vDash \psi$ iff $\sigma' \vDash \psi$.*

Definition 4 (Stuttering bisimilarity [3, 7]). *Given two LTSs $TS = \langle \mathcal{S}, A, \rightarrow \rangle$ and $TS' = \langle \mathcal{S}', A', \rightarrow' \rangle$ with initial states s_0 and s'_0, TS and TS' are said to be stuttering bisimilar w.r.t some set Π of atomic propositions and two labeling functions $L : \mathcal{S} \rightarrow 2^{\Pi}$ and $L' : \mathcal{S}' \rightarrow 2^{\Pi}$, if there exists a stuttering bisimulation $\mathcal{R}$ such that $s_0 \mathcal{R} s'_0$. A stuttering bisimulation $\mathcal{R}$ is a relation on $\mathcal{S} \times \mathcal{S}'$ such that whenever $s_1 \mathcal{R} s'_1$ then:*

1. *$L(s_1) = L'(s'_1)$,*
2. *for each maximal path $\lambda = s_1 s_2 \cdots \in \mathcal{S}^{\star}$ where $s_1 \rightarrow s_2 \rightarrow \ldots$ there exists a maximal path $\lambda' = s'_1 s'_2 \cdots \in \mathcal{S}'^{\star}$ where $s'_1 \rightarrow' s'_2 \rightarrow' \ldots$ such that λ and λ' can be partitioned into consecutive subpaths $B_1 B_2 \ldots$ and $B'_1 B'_2 \ldots$, respectively and $s \mathcal{R} s'$ for every s in B_i and s' in B'_i, for all $i > 0$.*
3. *symmetrically, for each path λ' from s'_1 there exists a subpathwise matching path λ from s_1.*

Theorem 3 (Stutter invariance [3]). *Let s and s' be two states. If $s \mathcal{R} s'$ for some stuttering bisimulation $\mathcal{R}$, then $s \vDash \varphi$ iff $s' \vDash \varphi$ for every CTL*\X formula φ.*

2.2.5 Reachability and safety In many cases we are only interested in very simple properties. E.g. that the system eventually reaches some *good state*, or that it never reaches some *bad state*, like a deadlock. These properties are called reachability and safety properties (Reach), and consist of exactly one outermost EF and AG pairs of operator, respectively. Since $EF\varphi \equiv \neg AG\neg\varphi$, reachability and safety properties are duals. They are fragments of both fragments of CTL and LTL.

3 Petri nets

A multiset $A : S \to \mathbb{N}_0$ is a set where each element is also associated with a multiplicity. Let $\mathbf{MS}(S)$ be all multisets with elements from the set S. Given a multiset $A \in \mathbf{MS}(S)$, let the support set $\mathbf{Supp}(A) = \{s \in S \mid A(s) \geq 1\}$ be the elements of S with a multiplicity of at least 1 in A. We assume the standard operations on multisets, like cardinality $|\cdot|$, membership $\in$, summation $\uplus$, subtraction $\setminus$, multiplication with a scalar, and inclusion $\subseteq$. We will use the following notation to describe a multiset $A : S \to \mathbb{N}_0$:

$$\sum_{s \in S} A(s)'s$$

E.g. if $A(x) = 3$ and $A(y) = 1$, then $A = 3'x + 1'y$. Remark that $1'x + 1'x$ and $2'x$ denotes the same multiset.

3.1 Colors

In colored Petri nets, tokens have colors, which represent their type. A color is a tuple of integers and

$$U = \bigcup_{n \in \mathbb{N}} \mathbb{N}_0^n$$

is the universe of colors containing all colors. Singleton colors are said to be base colors and are contained in $\mathbb{N}_0$. Given $\alpha, \beta \in \mathbb{N}_0$, let $[\alpha, \beta] \subseteq \mathbb{N}_0$ be an integer range with integers between α and β (inclusive). A color type is a product $D = [\alpha_1, \beta_1] \times \cdots \times [\alpha_n, \beta_n]$ with $\alpha_i \leq \beta_i$ for all $i \in [1, n]$. E.g. if $D = [0, 5] \times [0, 5] \times [0, 10]$, then $\langle 4, 2, 10\rangle \in D$. Let $\mathbb{D}$ be a fixed finite set of color types. Singleton color types are denoted $C = [\alpha, \beta]$, and $\mathbb{C} = \{D \in \mathbb{D} \mid |D| = 1\}$ is the set of all singleton color types.

We also define *successor* and *predecessor* functions $succ_C : C \to C$ and $pred_C : C \to C$ for all $C \in \mathbb{C}$. Given $c \in [\alpha, \beta] = C$, these shift colors as follows:

$$succ_C(c) = \begin{cases} c + 1 & \textbf{if } c < \beta \\ \alpha & \textbf{if } c = \beta \end{cases}$$

$$pred_C(c) = \begin{cases} c - 1 & \textbf{if } c > \alpha \\ \beta & \textbf{if } c = \alpha \end{cases}$$

3.2 Variables and bindings

Let $\mathbf{Var}$ be a finite set of variables and let $\mathcal{Y} : \mathbf{Var} \to \mathbb{C}$ be a type assignment function that assigns a color type to each variable. Given a $C \in \mathbb{C}$ we use $\mathbf{Var}_C = \{x \in \mathbf{Var} \mid \mathcal{Y}(x) = C\}$ to denote the set of variables of type C. Variables are assigned to colors using a binding $b : \mathbf{Var} \rightharpoonup \mathbb{N}_0$ such that $b(x) \in \mathcal{Y}(x)$. Let $\mathbb{B}$ be the set of all bindings.

Color expressions The set of all color expressions with type $C \in \mathbb{C}$ is $\mathbb{T}_C$, where a color expression $\tau_C \in \mathbb{T}_C$ is defined as:

$$\tau_C ::= c \mid x \mid \tau_C\texttt{++} \mid \tau_C\texttt{--}$$

with $c \in C$ and $x \in \mathbf{Var}_C$.

The semantics of color expressions are given by the function $[\![\cdot]\!]_{\mathbb{T}_C} : \mathbb{T}_C \times \mathbb{B} \to C$ which is defined inductively as:

$$\begin{aligned}
&[\![c, b]\!]_{\mathbb{T}_C} = c && \text{(constant)}\\
&[\![x, b]\!]_{\mathbb{T}_C} = b(x) && \text{(variable)}\\
&[\![\tau_C\texttt{++}, b]\!]_{\mathbb{T}_C} = succ_C([\![\tau_C, b]\!]_{\mathbb{T}_C}) && \text{(successor)}\\
&[\![\tau_C\texttt{--}, b]\!]_{\mathbb{T}_C} = pred_C([\![\tau_C, b]\!]_{\mathbb{T}_C}) && \text{(predecessor)}
\end{aligned}$$

The binding b must bind the relevant variables of course.

We also let $\mathbf{Vars}(\tau_C)$ denote all variables in τ_C, i.e.:

$$\begin{aligned}
&\mathbf{Vars}(c) = \emptyset && \text{(constant)}\\
&\mathbf{Vars}(x) = \{x\} && \text{(variable)}\\
&\mathbf{Vars}(\tau_C\texttt{++}) = \mathbf{Vars}(\tau_C) && \text{(successor)}\\
&\mathbf{Vars}(\tau_C\texttt{--}) = \mathbf{Vars}(\tau_C) && \text{(predecessor)}
\end{aligned}$$

Guard expressions Let Γ be a set of guards that evaluates to true or false when given a binding. A guard γ is defined inductively using the following syntax:

$$\gamma ::= \top \mid \bot \mid \neg\gamma_1 \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1 \vee \gamma_2 \mid \tau_C^1 \bowtie \tau_C^2$$

where $\tau_C^1, \tau_C^2 \in \mathbb{T}_C$ and $\bowtie \in \{<, \leq, =, \geq, >, \neq\}$.

The semantics of guards are given by the function $[\![\cdot]\!]_\Gamma : \Gamma \times \mathbb{B} \to \{\top, \bot\}$ which is defined inductively as:

$$\begin{aligned}
&[\![\top, b]\!]_\Gamma = \top && \text{(true)}\\
&[\![\bot, b]\!]_\Gamma = \bot && \text{(false)}\\
&[\![\neg\gamma, b]\!]_\Gamma = \neg[\![\gamma, b]\!]_\Gamma && \text{(negation)}\\
&[\![\gamma_1 \wedge \gamma_2, b]\!]_\Gamma = [\![\gamma_1, b]\!]_\Gamma \wedge [\![\gamma_2, b]\!]_\Gamma && \text{(and)}\\
&[\![\gamma_1 \vee \gamma_2, b]\!]_\Gamma = [\![\gamma_1, b]\!]_\Gamma \vee [\![\gamma_2, b]\!]_\Gamma && \text{(or)}\\
&[\![\tau_C^1 \bowtie \tau_C^2, b]\!]_\Gamma = [\![\tau_C^1, b]\!]_{\mathbb{T}_C} \bowtie [\![\tau_C^2, b]\!]_{\mathbb{T}_C} && \text{(comparison)}
\end{aligned}$$

When $[\![\gamma, b]\!]_\Gamma = \top$ for some guard expression γ and a binding b, we say that b is a valid binding for γ.

Finally, $\mathbf{Vars}(\gamma)$ denotes all variables in γ, i.e.:

$$\begin{aligned}
&\mathbf{Vars}(\top) = \emptyset && \text{(true)}\\
&\mathbf{Vars}(\bot) = \emptyset && \text{(false)}\\
&\mathbf{Vars}(\neg\gamma) = \mathbf{Vars}(\gamma) && \text{(negation)}\\
&\mathbf{Vars}(\gamma_1 \wedge \gamma_2) = \mathbf{Vars}(\gamma_1) \cup \mathbf{Vars}(\gamma_2) && \text{(and)}\\
&\mathbf{Vars}(\gamma_1 \vee \gamma_2) = \mathbf{Vars}(\gamma_1) \cup \mathbf{Vars}(\gamma_2) && \text{(lor)}\\
&\mathbf{Vars}(\tau_{\mathcal{C}}^1 \bowtie \tau_{\mathcal{C}}^2) = \mathbf{Vars}(\tau_{\mathcal{C}}^1) \cup \mathbf{Vars}(\tau_{\mathcal{C}}^2) && \text{(comparison)}
\end{aligned}$$

3.3 Colored Petri nets

Now that we have defined all the color components, we will now define a colored Petri net as a bipartite graph of places and transitions including these colors and expressions.

Definition 5 (Colored Petri net). *A colored Petri net (CPN) is a 7-tuple $\langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G\rangle$, where*

- *P is a non-empty finite set of places,*
- *T is a finite set of transitions,*
- *$\mathcal{X} : P \to \mathbb{D}$ is the color domain function assigning color types to places,*
- *$\boxminus : (P \times T) \to \boldsymbol{MS}(\boldsymbol{Var} \times \cdots \times \boldsymbol{Var})$ and $\boxplus : (T \times P) \to \boldsymbol{MS}(\boldsymbol{Var} \times \cdots \times \boldsymbol{Var})$ are the pre- and post-incident weight functions such that if $p \in P$ and $t \in T$ then*

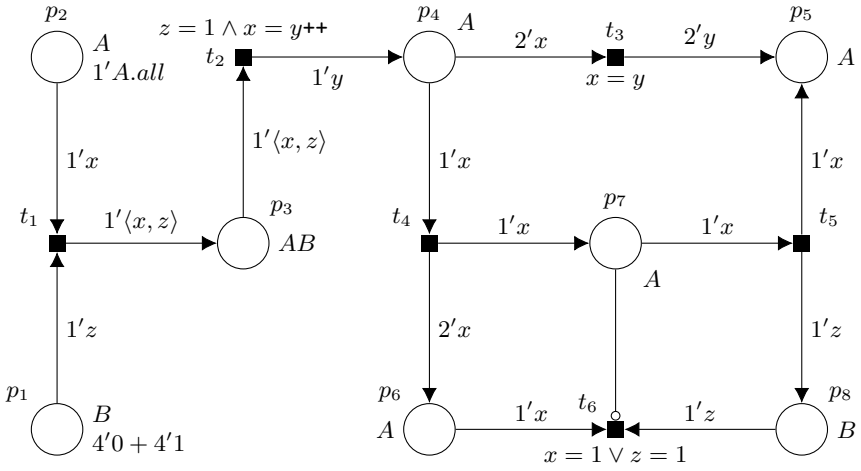
$$\begin{aligned}
&\forall\langle x_1, \ldots, x_n\rangle \in \boxminus(p,t).\mathcal{Y}(x_1) \times \cdots \times \mathcal{Y}(x_n) = \mathcal{X}(p), \text{and}\\
&\forall\langle x_1, \ldots, x_n\rangle \in \boxplus(p,t).\mathcal{Y}(x_1) \times \cdots \times \mathcal{Y}(x_n) = \mathcal{X}(p)
\end{aligned}$$

- *$I : P \times T \to \mathbb{N}_\infty$ is the inhibitor weight function,*
- *$G : T \to \Gamma$ assigns guards to transitions such that the guard only contains variables that are on the associated arcs to the transition. I.e. let $\boldsymbol{Vars}(t)$ be the set of all variables that appear in the tuples of the multiset $\boxminus(p,t)$ or $\boxplus(t,p)$ for any $p \in P$, then $\boldsymbol{Vars}(G(t)) \subseteq \boldsymbol{Vars}(t)$.*

Remark 1. Our definition of CPNs differs slightly from the traditional definition. Without loss of generality, we do not allow expressions on arcs, only a multiset of variables. We instead assume such expressions are given in the guard instead.

A *marking* is a function $M : P \to \mathbf{MS}(U)$ that assigns a number of colored tokens to each place such that $M(p) \in \mathbf{MS}(\mathcal{X}(p))$ for all $p \in P$. The initial marking of a CPN is denoted M_0, and the set of all markings of a CPN N is denoted $\mathcal{M}(N)$.

Figure 2 shows a CPN using circles for places, filled squares for transitions, arrows for arcs representing the pre- and post-incidental weight functions, and



Color types:	**Var types**:
$A = [1, 5]$	$\mathcal{Y}(x) = A$
$B = [0, 1]$	$\mathcal{Y}(y) = A$
$AB = A \times B$	$\mathcal{Y}(z) = B$

Figure 2: Example CPN

circle-head arrows for inhibitor arcs. We have that $\mathbb{D} = \{A, B, AB\}$ and $\mathbf{Var} = \{x, y, z\}$. The type of a place, and the initial marking is written next to the place. Note that $1'A.all$ is a conventional shorthand for $\sum_{c \in A} 1'c$. Lastly, guards that are not $\top$ are also written next to the relevant transition.

For a transition $t \in T$ to *fire*, it requires a binding $b : \mathbf{Vars}(t) \to \mathbb{N}_0$ that binds exactly the relevant variables. Let $B(t)$ denote all bindings for transition t that binds exactly $\mathbf{Vars}(t)$. We also define a set of all valid bindings for transition t as $\vec{B}(t) = \{b \in B(t) \mid [\![G(t), b]\!]_\Gamma = \top\}$. Given a binding $b \in B(t)$ and a multiset of variables like $\boxminus(p, t)$, we will write $b(\boxminus(p, t))$ to denote a multiset of colors created by summing the multiplicity of variables that are mapped to the same color. Specifically, using multiset sum notation:

$$b(\boxminus(p, t)) = \sum_{x \in \boxminus(p,t)} (\boxminus(p, t)(x))'b(x)$$

A marking M can transition to marking M' by firing transition $t \in T$ using binding $b \in \vec{B}(t)$, denoted $M \xrightarrow{t,b} M'$, if $[\![G(t), b]\!]_\Gamma = \top$ and $\forall p \in P.b(\boxminus(p, t)) \subseteq M(p) \wedge |M(p)| < I(p, t)$ such that $\forall p \in P$ we have

$$M'(p) = M(p) \uplus b(\boxplus(t, p)) \setminus b(\boxminus(p, t))$$

We say that transition t is *enabled* in marking M, if there exists a binding b such that $M \xrightarrow{t,b}$. We will write $M \xrightarrow{t}$ if there exists a binding $b \in B(t)$ such

that $M \xrightarrow{t,b}$. A CPN $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ thus encodes an LTS $TS(N) = \langle \mathcal{S}, A, \rightarrow \rangle$ where $S = \mathcal{M}(N)$, $A = T \times \mathbb{B}$, and $M \xrightarrow{t,b} M'$ is defined as above.

We also define the following notation:

- the *pre set* (producers) of a place $p \in P$ as ${}^\bullet p = \{t \in T \mid \boxplus(t,p) \neq \emptyset\}$,
- the *post set* (consumers) of a place $p \in P$ as $p^\bullet = \{t \in T \mid \boxminus(p,t) \neq \emptyset\}$,
- the *pre set* of a transition $t \in T$ as ${}^\bullet t = \{p \in P \mid \boxminus(p,t) \neq \emptyset\}$,
- the *post set* of a transition $t \in T$ as $t^\bullet = \{p \in P \mid \boxplus(t,p) \neq \emptyset\}$,
- the *inhibiting pre set* of a transition $t \in T$ as ${}^\circ t = \{p \in P \mid I(p,t) < \infty\}$,
- the *inhibited post set* of a place $p \in P$ as $p^\circ = \{t \in T \mid I(p,t) < \infty\}$,
- the *increasing pre set* for a place $p \in P$ as ${}^{\boxplus} p = \{t \in {}^\bullet p \mid |\boxplus(t,p)| > |\boxminus(p,t)|\}$,
- the *decreasing post set* for a place $p \in P$ as $p^{\boxminus} = \{t \in p^\bullet \mid |\boxplus(t,p)| < |\boxminus(p,t)|\}$,
- the *transmuting pre set* of a place $p \in P$ as ${}^\nabla p = \{t \in {}^\bullet p \mid \boxplus(t,p) \neq \boxminus(p,t)\}$,
- the *transmuted post set* of a transition $t \in T$ as $t^\nabla = \{p \in t^\bullet \mid \boxplus(t,p) \neq \boxminus(p,t)\}$.

For a set X of either places or transitions, we extend the notation as ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$, and likewise for the other operators.

3.4 P/T nets

Consider a colored Petri net $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ where there is just one colour, and as such $\mathcal{X}(p) = \{\bullet\}$ for all $p \in P$. This implies that $b(x) = \bullet$ for all variables $x \in \mathbf{Var}_{\{\bullet\}}$ and all bindings $b \in \vec{B}(t)$ where $t \in T$. Hence, all variables are equivalent and there might as well only be one variable x. However, this implies that there is only one binding too. Since there is only one binding, all guards $G(t) = \top$ or semantically equivalent for all transitions $t \in T$, otherwise t has no valid bindings and is superfluous. Moreover, we have that for all $p \in P$ there exists a $w_1 \in \mathbb{N}_0$ such that $M_0(p) = w_1{}'\bullet$, and for all $p \in P$ and $t \in T$ there exists $w_2, w_3 \in \mathbb{N}_0$ such that $\boxminus(p,t) = w_2{}'x$ and $\boxplus(t,p) = w_3{}'x$. In other words, the initial marking and arcs are defined by their cardinality alone. Such a colored Petri net is also called a place/transition net (P/T net) or simply a Petri net. Since there is only one color, variables, bindings, and guards are unnecessary. From the observations above, we define a P/T net as follows from now on.

Definition 6. *A place/transition net (P/T net) is a CPN* $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ *where*

- $\mathcal{X}(p) = \{\bullet\}$ *for all* $p \in P$,
- $G(t) = \top$ *for all* $t \in T$,
- $\boldsymbol{Var}_{\{\bullet\}} = \{x\}$ *and for all* $p \in P$ *and all* $t \in T$ *there exists weights* $w_1, w_2 \in \mathbb{N}_0$ *such that* $\boxminus(p,t) = w_1{}'x$ *and* $\boxplus(t,p) = w_2{}'x$.

In the context of P/T nets, a marking M is treated as a function $P \rightarrow \mathbb{N}_0$ that returns the number of tokens in the given place. Similarly, the pre- and post

incident weight functions, $\boxminus$ and $\boxplus$, are respectively treated as functions of the type $(P \times T) \to \mathbb{N}_0$ and $(T \times P) \to \mathbb{N}_0$, that return the weight of the given arc.

A marking M can transition to M' using transition $t \in T$, denoted $M \xrightarrow{t} M'$, if $\forall p \in P. \boxminus (p,t) \leq M(p) < I(p,t)$ such that $\forall p \in P$ we have $M'(p) = M(p) + \boxplus(t,p) - \boxminus(p,t)$. Note the absence of bindings.

All CPNs can be unfolded to an isomorphic P/T net [19]. The unfolded net will typically be exponentially larger than the original CPN. Hence, performing structural reductions on the CPN can lead to a much smaller unfolded net, and reduce overhead in unfolding and subsequent model checking techniques. Unfolding is desirable, since there are many verification techniques developed for P/T nets such as linear state equations, marked graphs, stubborn reductions, and heuristics [23, 5, 18]

4 Strutural reductions

A structural reduction is a modification of the structure of a model, such that the resulting model is either smaller or simpler. However, the reduction must also preserve any properties of interest to be useful. An analysis of local structures is therefore needed to check if the preconditions for a reduction is satisfied.

When performing a structural reduction on a Petri net, we cannot remove places and transitions relevant for the atomic propositions of the property in question. Therefore, given a CTL* formula φ we use the function $\mathit{places}(\varphi)$ to represent all the places mentioned in the formula φ. Specifically, the function *places* is defined inductively as:

$$
\begin{aligned}
&\mathit{places}(\top) = \emptyset \\
&\mathit{places}(\alpha M \bowtie k) = \{p \in P \mid \alpha(p) \neq 0\} \\
&\mathit{places}(\mathit{en}(t)) = \emptyset \\
&\mathit{places}(\mathit{deadlock}) = \emptyset \\
&\mathit{places}(\neg\varphi) = \mathit{places}(\varphi) \\
&\mathit{places}(\varphi_1 \wedge \varphi_2) = \mathit{places}(\varphi_1) \cup \mathit{places}(\varphi_2) \\
&\mathit{places}(A\psi) = \mathit{places}(\psi) \\
&\mathit{places}(E\psi) = \mathit{places}(\psi) \\
&\mathit{places}(\neg\psi) = \mathit{places}(\psi) \\
&\mathit{places}(\psi_1 \wedge \psi_2) = \mathit{places}(\psi_1) \cup \mathit{places}(\psi_2) \\
&\mathit{places}(X\psi) = \mathit{places}(\psi) \\
&\mathit{places}(F\psi) = \mathit{places}(\psi) \\
&\mathit{places}(G\psi) = \mathit{places}(\psi) \\
&\mathit{places}([\psi_1 U \psi_2]) = \mathit{places}(\psi_1) \cup \mathit{places}(\psi_2)
\end{aligned}
$$

Similarly, the function $transitions(\varphi)$ returns all transitions mentioned in φ. Specifically:

$$
\begin{aligned}
&transitions(\top) = \emptyset \\
&transitions(\alpha M \bowtie k) = \emptyset \\
&transitions(en(t)) = \{t\} \\
&transitions(deadlock) = \emptyset \\
&transitions(\neg\varphi) = transitions(\varphi) \\
&transitions(\varphi_1 \wedge \varphi_2) = transitions(\varphi_1) \cup transitions(\varphi_2) \\
&transitions(A\psi) = transitions(\psi) \\
&transitions(E\psi) = transitions(\psi) \\
&transitions(\neg\psi) = transitions(\psi) \\
&transitions(\psi_1 \wedge \psi_2) = transitions(\psi_1) \cup transitions(\psi_2) \\
&transitions(X\psi) = transitions(\psi) \\
&transitions(F\psi) = transitions(\psi) \\
&transitions(G\psi) = transitions(\psi) \\
&transitions([\psi_1 U \psi_2]) = transitions(\psi_1) \cup transitions(\psi_2)
\end{aligned}
$$

Definition 7 (Invisibility). *If $\varphi \in CTL^*\backslash X$ then a transition $t \in T$ is said to be* invisible *to φ if*

- *for all $p \in places(\varphi)$ we have that $|\boxminus(p,t)| = |\boxplus(t,p)|$,*
- *for all $p \in {}^\bullet transitions(\varphi)$ we have that $\boxminus(p,t) \subseteq \boxplus(t,p)$, and*
- *for all $p \in {}^\circ transitions(\varphi)$ we have that $|\boxminus(p,t)| \geq |\boxplus(t,p)|$.*

Hereby, firing t does not affect the satisfaction of the atomic propositions in φ.

Our structural reductions will be presented as rules in Section 5 and Section 6. Each rule (except Rule I) consists of a list of precondition which must hold, and a list of updates which describe the modifications that are made to the net on application of the rule. While the preconditions can be checked in any order, some updates must happen sequentially which is emphasized by the word "then" between two items in the update list. Rule I is instead presented as an algorithm. Each rule will be proven correct for CTL* or some of its fragments. We define the correctness of a rule below:

Definition 8 (Correctness of Rule X for logic L).
Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G\rangle$ be a Petri net and let $M_0 \in \mathcal{M}(N)$ be its initial marking. Let $N^\varphi = \langle P^\varphi, T^\varphi, W^\varphi_{\boxminus}, W^\varphi_{\boxplus}, I^\varphi\rangle$ with initial marking $M_0^\varphi \in \mathcal{M}(N^\varphi)$ be the Petri net obtained by applying Rule X on N once for some formula φ of logic L. Rule X is correct for logic L if:

$$M_0 \vDash \varphi \text{ in } N \text{ iff } M_0^\varphi \vDash \varphi \text{ in } N^\varphi \text{ for all } \varphi \in L.$$

When a rule is correct for a class of logic, it is also correct for its fragments.

Unless otherwise specified, a CPN presented as $N' = \langle P', T', \mathcal{X}', \boxminus', \boxplus', I', G' \rangle$ refers to the CPN $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ reduced using one application of the discussed rule.

4.1 Structural reduction categories

We categorise our proposed structural reductions into three overall categories. These categories are redundant structures, parallel structures, and agglomerations.

4.1.1 Redundant structures Redundant structures do not provide the net with interesting behaviour. They can be places that never disable their consumers, transitions that are never enabled, or transitions which can only be fired in one way. These structures are often completely redundant and unnecessary for the properties and can thus be removed. We define four rules, Rules E, F, I, and Q in Section 6.3, 6.4, 6.5, and 6.7 that focus on removing these redundant structures. Rule I is more specialized and focuses on structures that cannot affect the places mentioned in the property.

4.1.2 Parallel structures When two places or transitions exist in the net such that one shares all of its pre- and post-set with the other, these can be considered parallel to each other. See Figure 3 for an example of parallel places. Here p_1 always imposes more strict conditions on t_1, and as such we can remove p_2 as it is redundant. Similar cases can be made for parallel transitions. Rule C and D, introduced in Section 6.1 and 6.2 remove parallel structures in the net.

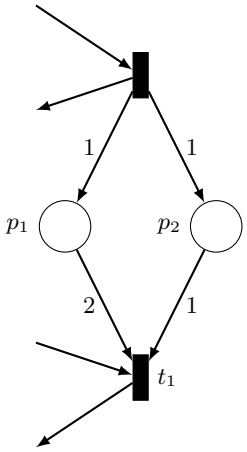


Figure 3: Places in parallel

4.1.3 Agglomerations The main idea in agglomerations is to combine transitions which must also be fired in sequence. See Figure 4 for an example of a simple agglomeration. An agglomeration does not necessarily result in a smaller net depending on the number of sequences possible. However, interleavings in the underlying concurrent behaviour is removed, resulting in a smaller state space. Our Rule S for P/T nets introduced in Section 5.2 and later generalized to CPNs in Section 6.6 is an atomic free agglomeration only removing a single consumer at a time, which makes the rule more general than a free agglomeration [28].

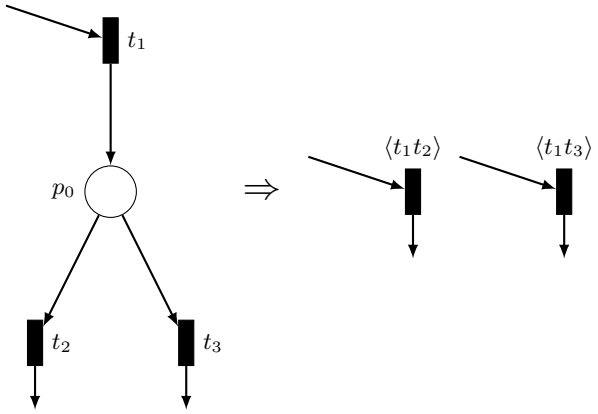


Figure 4: Example of agglomeration

4.2 Benchmark and experimental setup

In Section 5 and 6 we present our structural reductions for P/T nets and CPNs, respectively. At the end of these sections, we present the results of applying the presented rules. Therefore, we present our experimental setup now, as this is used in both of these result sections.

We implement the reductions rules described in the following sections in `verifypn` [17, 4] which is the verification engine used in the Petri net verification tool TAPAAL [8]. The engine `verifypn` is written in C++ and prior to this thesis, had no structural reductions for CPNs. Our presented rules for P/T nets has been implemented along the existing structural rules. The modified version of `verifypn` can be found on GitHub. All results from testing, and all generated figures can also be found on GitHub.

We experimentally test the performance of our rules using the CPN and P/T net models from the Model Checking Contest, MCC2021 [21]. For the P/T rules in Section 5 we use the 1181 P/T nets, while for the CPN rules in Section 6 we use the 230 CPNs. We use both the cardinality and fireability queries of the reachability, LTL, and CTL categories. This gives us a test suite of 113.376

queries for P/T nets and 22.080 queries for CPNs. The hardware we use for testing is the DEIS MCC, which is a computer cluster at Aalborg University, configured for repeatability and stability and memory-intensive computations. Each query is allocated a maximum of 15GB memory and 2 cores on a Xeon e5 2680 CPU. If the memory limit is reached, the query is automatically terminated.

Our experiments are compared to a *base* experiment which uses `verifypn` without our rules presented in this thesis, but with the existing P/T reduction rules enabled. For all queries, we measure multiple metrics:

- *colored reduce time* is the time spent applying colored structural reductions (time limit of 4 seconds).
- *unfolding time* is the time spent unfolding the net.
- *reduce time* is the time spent applying structural reductions on the unfolded P/T net (time limit of 30 seconds).
- *verification time* is the time spent on verification.
- *total time* is the sum of the colored reduce time, unfolding time, reduce time, and verification time.
- *applications of X* is the number of times rule X was applied.
- *size of the net* is the size given by adding the number of places and transitions together. This is measured before, and after the reduction and unfolding phases.
- *answers* is the number of answers found within the time and memory limit.

If a query reaches the time limit of any reduction phase, then the engine proceeds to the next phase. If the answer to the query is not found within 4 minutes, the query is also terminated. Separately, if possible within 2 minutes, we measure the *state space size*, i.e. the number of states in the final unfolded and reduced P/T net.

We run the experiments with different search strategies during verification of the property, including depth-first search, random depth-first search, and a heuristic search using the manhattan distance to a marking satisfying the property [16]. However, we find that the effect of using our rules does not vary much across different search strategies. Therefore, we present only the results using depth-first search, as this generally performed the best amongst the tested search strategies, both with and without our rules applied.

In addition to testing on the models mentioned, we have also created a modified version of all P/T nets and CPNs in the MCC2021 with inserted inhibitor arcs. We do this in order to test the correctness of our presented rules on nets with inhibitor arcs. The nets have been modified, by inserting a number of inhibitor arcs proportional to the size of the net between random places and transitions, and their weight was set to either 1 or 2, chosen randomly. Because of this randomness, the modified models become nonsensical, but still useful for verifying the correctness of our rules.

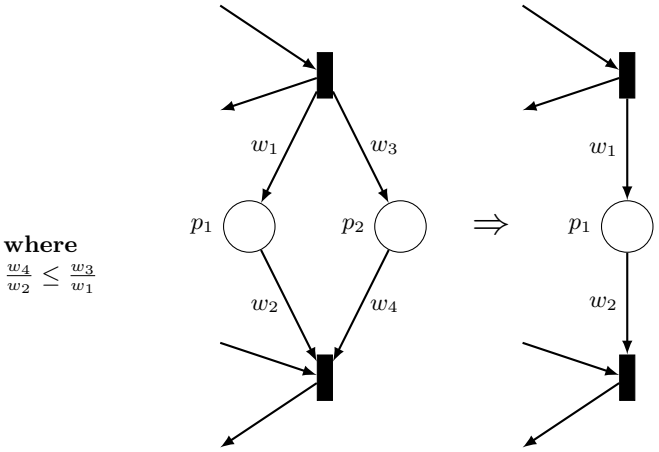
The results of the experiments using our P/T net rules are presented in Section 5.3. The results of the experiments using CPN our rules are presented in Section 6.8. In Section 7 we discuss our overall findings.

5 Structural reductions for P/T nets

In this section we present our structural reduction rules for P/T nets.

5.1 Rule C: Parallel Places (P/T)

If a pair of places are parallel and one may accumulate tokens because the other always limits its consumers, then Rule C will remove the one that is not the limiting factor. See Figure 5. By convention $\min \emptyset = -\infty$ and $\max \emptyset = \infty$. The fraction d describes how quickly tokens can be consumed from p_2 compared to p_1, while f describes how slowly tokens can be fed to p_2 compared to p_1. If $d \leq f$



Precondition	Update
Fix places p_1 and p_2 s.t.: C1) $p_2 \notin places(\varphi)$ C2) $p_2^\circ = \emptyset$ C3) $p_1^\bullet \neq \emptyset$ C4) $p_1^\bullet \supseteq p_2^\bullet$ C5) ${}^\bullet p_1 \subseteq {}^\bullet p_2$ C6) $M(p_2) \geq M(p_1) \cdot d$ C7) $d \leq f$ where $d = \max_{t \in p_1^\bullet} \frac{\boxminus(p_2,t)}{\boxminus(p_1,t)}$ $f = \min_{t \in {}^\bullet p_1} \frac{\boxplus(t,p_2)}{\boxplus(t,p_1)}$	UC1) remove p_2

Figure 5: Rule C: Parallel places (P/T)

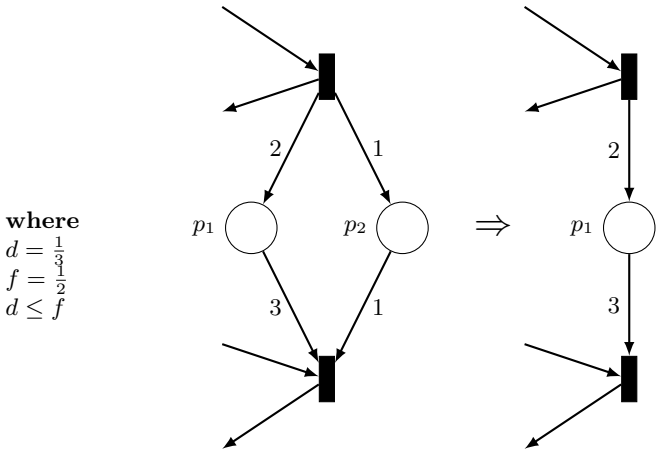


Figure 6: A reduction that our Rule C can perform, but Rule C from [5] cannot

then p_2 is always fed faster than it is emptied compared to p_1, which means p_2 can be removed, since it will always be p_1 which is missing tokens and disables their consumers.

Rule C is a generalization of Rule C from [5]. Our Rule C is able to reduce the net shown in Figure 6 while Rule C by Bønneland et al. cannot.

Theorem 4. *The P/T net Rule C shown in Figure 5 is correct for CTL*.*

Proof. Using contradiction we will show that a net reduced by Rule C is bisimilar to the original net since no behaviour is removed. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a P/T net with initial marking M_0 and let $\varphi \in$ CTL*. Let N' and M_0' be the P/T net and initial marking after applying Rule C once on N. Let M be a reachable marking s.t. $M_0 \rightarrow^* M$. To argue for the correctness of Rule C, we will show that if any $t \in T$ is disabled due to the number of tokens in p_2 in marking M, then it is also disabled due to the number of tokens in p_1. In other words, that $M(p_2) < \boxminus(p_2, t)$ implies $M(p_1) < \boxminus(p_1, t)$ for all $t \in T$. To show this, we must realize that in conjunction with C4 and C5, that d is the fastest rate at which p_2 can be emptied compared to p_1, and conversely, that f is the slowest rate at which p_2 can be fed compared to p_1. Both d and f are well defined by C3 and that by convention that $\min \emptyset = -\infty$.

We will now show that $M(p_2) < \boxminus(p_2, t)$ implies $M(p_1) < \boxminus(p_1, t)$ by contradiction. Assume there exist a transition $t \in T$ and a reachable marking M such that $M_0 \rightarrow^* M$ where $M(p_1) \geq \boxminus(p_1, t)$ and $M(p_2) < \boxminus(p_2, t)$. By the definition of d, this means that $M(p_2)/M(p_1) < d$. By C6, we have that $M_0(p_2)/M_0(p_1) \geq d$ and thus $M_0 \neq M$. Since M where $M(p_2)/M(p_1) < d$ is reachable there must exist a transition $t' \in T$ that either:

1. consumes at least d times more tokens from p_2 than from p_1, i.e. $\boxminus(p_2, t')/\boxminus(p_1, t') > d$, or

2. produces at most d times more tokens to p_1 than to p_2, i.e. $\boxplus(t', p_2) / \boxplus (t', p_1) < d$.

Case 1. is impossible by C4 and the definition of d. Case 2. is also impossible due to the C5, C7, and the definition of f. Alas we have a contradiction.

In combination with C2, we can conclude that N' must be bisimilar, and hence Rule C is correct for CTL* by Theorem 1.

□

5.2 Rule S: Atomic free agglomeration

An agglomeration aims to replace sequences of transition firings with new transitions which are for all query-relevant purposes identical to firing the original sequence in immediate succession. An agglomeration is centered around a single place, whose producers and consumers are the transitions that will be merged. This place will be referred to as p_0. For an illustration of this refer to Figure 4.

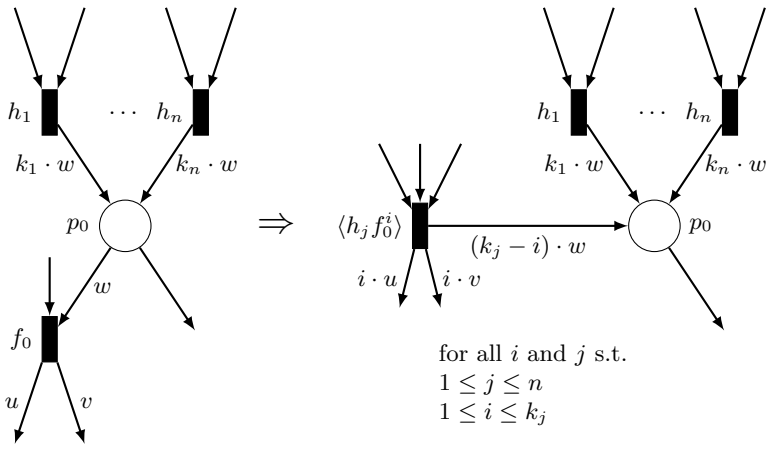
In a pre agglomeration, the p_0 has a disjoint pre set and post set, and p_0 is the only place produced to by its pre set. Hence, the firing of any consumer $f \in p_0^\bullet$ must be preceded by exactly one producer $h \in {}^\bullet p_0$ and the firing of h can be delayed until needed. A pre agglomeration introduces this delay by merging every pair h and f, which in turn removes interleavings in the net. See Appendix A for the definition of Rule T which is a pre agglomeration.

A free agglomeration [28] is a variant of a pre agglomeration, which does not require the pre set of h to only have h as their post set, but as a result only preserves reachability properties without deadlock.

Our Rule S, described in Figure 7, is an atomic free agglomeration. By atomic, we mean, that we do not have to merge every consumer $f \in p_0^\bullet$ with every producer $h \in {}^\bullet p$. Instead, we find one $f_0 \in p_0^\bullet$ which, after firing any $h \in {}^\bullet p_0$, can be fired $k \geq 1$ times with no leftover tokens in p_0. Rule S removes the original f_0 and creates for every $i \in [1, k]$ a transition equivalent to firing h and then f_0 i times. The intuition is, that every firing of h is used to fire f_0 a number of times, and since a merged transition is created for every possible number of times, no behaviour is lost, and thus reachability properties are preserved. Safety properties are preserved since the pre set of h is unobserved, i.e. ${}^\bullet h \cap places(\varphi) = \emptyset$.

Rule S is a generalization of the free agglomerations by Thierry-Mieg in [28] in the following ways:

- Atomic: Rule S only agglomerates one consumer at a time and is still applicable even if not all consumers can be agglomerated.
- Weighted: Rule S allows weights on the arcs to and from p_0. It also allows the p_0 to have an initial marking greater than 0, as long as it does not initially enable consumers to be agglomerated.
- k-scaled: The pre- and post arcs of the agglomerated place can have different weights. All weights on the pre arcs must be a multiple of the weight of the consumer arc. Rule S produces a transition for every possible number of



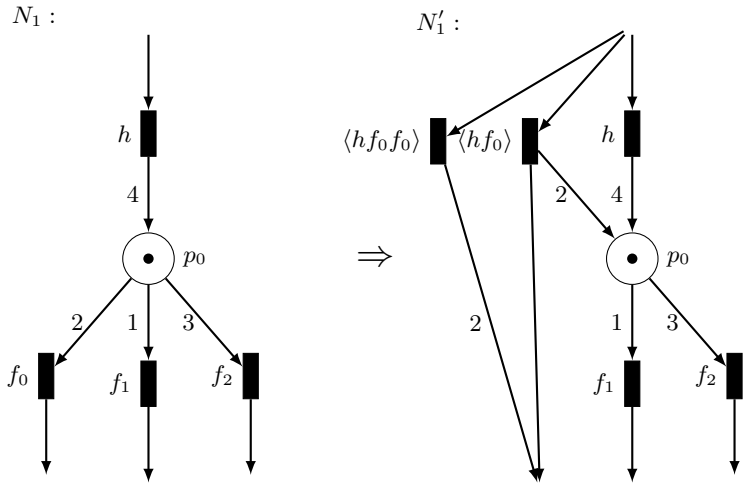
Precondition	Update
Fix place p_0 and transition f_0 s.t.: S1) $\{p_0\} \cap places(\varphi) = \emptyset$ S2) $(f_0 \cup {}^\bullet p_0) \cap transitions(\varphi) = \emptyset$ S3) $M_0(p_0) < \boxminus(p_0, f_0)$ S4) ${}^\bullet p_0 \cap p_0^\bullet = \emptyset$ S5) $f_0 \in p_0^\bullet$ and for all $h \in {}^\bullet p_0$ there exists a $k \in \mathbb{N}$ s.t.: S6) $h^\bullet = \{p_0\}$ S7) ${}^\bullet h \cap places(\varphi) = \emptyset$ S8) $p_0^\circ = {}^\circ h = ({}^\bullet h)^\circ = \emptyset$ S9) $\boxplus(h, p_0) = k \cdot \boxminus(p_0, f_0)$ S10) $k > 1 \implies (f_0^\bullet)^\circ = \emptyset$ S11) $k > 1 \implies {}^\bullet f_0 = \{p_0\}$	Create transition $\langle h f_0^i \rangle$ for all $i \in [1, k]$, for $k = \boxplus(h, p_0) / \boxminus(p_0, f_0)$, for all $h \in {}^\bullet p_0$. For each such transition: US1) $\boxplus(\langle h f_0^i \rangle, p_0) = \boxplus(h, p_0) - i \cdot \boxminus(p, f_0)$ and for all $p \in P \setminus \{p_0\}$: US2) $\boxminus(p, \langle h f_0^i \rangle) = \boxminus(p, h) \uplus \boxminus(p, f_0)$ US3) $\boxplus(\langle h f_0^i \rangle, p) = i \cdot \boxplus(f_0, p)$ US4) $I(p, \langle h f_0^i \rangle) = I(p, f_0)$ and US5) Remove f_0 US6) Then, if $p_0^\bullet = \emptyset$, remove p_0 and all transitions in ${}^\bullet p_0 \setminus transitions(\varphi)$

Figure 7: Rule S: Atomic free agglomeration

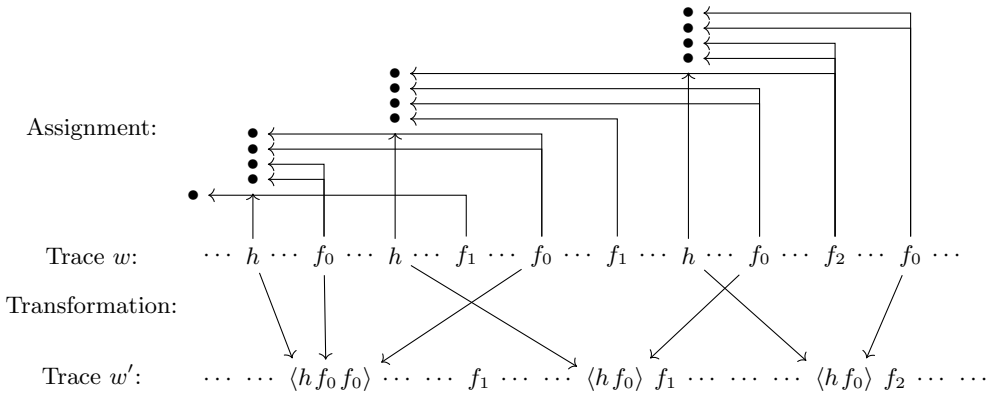
times the consumer could have been fired as a consequence of firing the given producer once. Since every variation is created, no reachability is altered aside from deadlock.

Ultimately, multiple applications of Rule S can accomplish the same reduction as the free agglomeration as described in [28], but Rule S is also more applicable.

Before proving that Rule S is correct for reachability properties without the deadlock proposition, we give the intuition for how an equivalent trace is found in the reduced net. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a P/T net and let N' be



(a)



(b)

Figure 8: Example of an assignment and transformation of a trace. (a) shows a P/T net N_1 and N_1' where N_1' is N_1 after one application of S agglomerating f_0. (b) shows a trace w fireable in N_1 and how it is transformed into the equivalent trace w' which is fireable in N_1'.

N after one application of Rule S. Let $w \in T^*$ be a trace which is fireable in N. If w contains one or more firings of f_0 then the reachability-equivalent trace $w' \in T'*$ in N' contains one or more firings of $\langle h f_0^i \rangle$ for some $i > 0$. By S6 and S8, we can delay firings of $h \in {}^\bullet p_0$ until tokens produced by it are needed in p_0. And if $k > 1$ then we can also advance the firings of f_0. However, f_0 may not be the only consumer of p_0, so which firings of h and f_0 can be merged is not trivial to find. We have to consider *when* the tokens are produced and which firings of

$f \in p_0^\bullet$ consumes them. Consider the example in Figure 8, which shows how a trace $w \in T^*$ is transformed to an equivalent trace $w' \in T'^*$. The transformation is based on an assignment which assigns each firing of $f \in p_0^\bullet$ to a set of tokens which it consumes, and firings f_0 are always assigned to tokens originating from a single firing of h. In the transformation, firings of h are delayed until they are needed, and each firing of f_0 is advanced to right after the h which produces the tokens f_0 is uniquely assigned to. Lastly, the firings of h and f_0 are merged to $\langle hf_0^i \rangle$. Such a transformation is always possible given the preconditions of Rule S and since w is fireable in N.

Theorem 5. *Rule S in Figure 7 is correct for reachability without deadlock*

Proof. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a P/T net with initial marking M_0 and let $\varphi \in$ Reach without deadlock. Let N' and M_0' be the P/T net and initial marking after applying Rule S once on N. We will now argue that any reachable marking M such that $M_0 \rightarrow^* M$ in N has a corresponding reachable marking M' such that $M_0' \rightarrow^* M'$ in N' where M and M' are equivalent w.r.t. the atomic propositions of φ, and conversely, every reachable marking M' in N' has an corresponding reachable marking M in N equivalent w.r.t. φ.

Let $w \in T^*$ be the trace such that $M_0 \xrightarrow{w} M$. By S5, every firing of $f \in p_0^\bullet$ in w consumes tokens from p_0, which were either there since the initial marking M_0 or produced by one or more producers in ${}^\bullet p_0$. To proceed, we identify tokens in p_0 based on when they were produced. That is, every token is associated with the position in the trace where it is created. By S3, we have that f_0 cannot be fired without firing an $h \in {}^\bullet p_0$ first, and by S9, every firing of f_0 can consume tokens which solely originate from one unique firing of an $h \in {}^\bullet p_0$. We must now realize, that we can associate every firing of an $f \in p_0^\bullet$ to a set of tokens which it consumes, such that every firing of f_0 is associated with a set of tokens which all originate from the same firing of some h. We call such an association between firings of f and its consumed tokens for an assignment. More than one of such assignments may exists, but we only need to consider one of them. Based on this assignment we will now transform w to a trace w'. The transformation satisfies that $M_0' \xrightarrow{w'} M'$ and M and M' are equivalent w.r.t. the atomic propositions of φ. The steps of the transformation are:

1. Delay the firing of every $h \in {}^\bullet p_0$ to right before the first firing of an $f \in p_0^\bullet$ which consumes tokens produced by the given firing of h according to the assignment.
2. Advance each firing of f_0 to right after the unique firing of $h \in {}^\bullet p_0$ which the tokens that f_0 consumes originate from according to the assignment.
3. Replace every occurrence of the subtrace hf_0^i with a firing of $\langle hf_0^i \rangle \in T'*$ where i is the greatest number such that the subtrace is not followed by additional firings of f_0.

We will now argue for the fireability of the resulting trace and the equivalence of the resulting state.

1. By S8 and S10, delaying the firing h does not affect which transitions are inhibited. By S6, only the transitions of $p_0^\bullet$ require tokens produced by h and thus the resulting trace is fireable and results in the same state.
2. Let $k = \boxplus(h, p_0)/\boxminus(p_0, f_0)$. If $k = 1$ then the relevant firing of h was moved to right before the given firing of f_0 in step 1 and no transformation occurs in this step. Otherwise, if $k > 1$, then by S10 and by S11, firing f_0 does not affect which transitions are inhibited and f_0 only consumes from p_0, which implies that the advancing of the firing of f_0 is possible without changing the resulting state.
3. By step 1 and 2, every firing of f_0 now immediately succeeds the firing of the $h \in {}^\bullet p_0$ which produces the tokens that the firings of f_0 consume. Let i be the number of firings of f_0 following a given h. By S9 and the definition of the assignment, we have that $i \leq k = \boxplus(h, p_0)/\boxminus(p_0, f_0)$. By US1, US2, US3, and US4, the reduced net has a transition $\langle h f_0^i \rangle$ which has the same effect as firing h followed by i firings of f_0.

This proves that every reachable marking M such that $M_0 \xrightarrow{*} M$ in N has a corresponding reachable marking M' such that $M'_0 \xrightarrow{*} M'$ in N' where M and M' are equivalent w.r.t. the atomic propositions of φ.

Now we prove it in the other direction. Let $w' \in T'^*$ be a trace such that $M'_0 \xrightarrow{w'} M'$. Let w be identical to w' but with every occurrence of $\langle h f_0^i \rangle$ replaced with $h f_0^i$. By US1, US2, US3, and US4, these subtraces have the same effect, and thus $M_0 \xrightarrow{w} M$ such that M and M' are equivalent w.r.t. the atomic propositions of φ, which concludes our proof. □

As Thierry-Mieg argues in [28], free agglomerations do not preserve deadlock properties, since it can remove deadlocks. However, our atomic free agglomeration Rule S can also introduce new deadlocks. An example of this is shown in Figure 9.

Our implementation of Rule S does not follow the definition above. Since the atomic free agglomeration Rule S and the pre agglomeration Rule T given in Appendix A share several preconditions, the implementation of Rule S falls back to a Rule T depending on what is possible given the net structure around p_0 and the query. This fallback also happens on categories for which Rule T is valid but Rule S is not.

5.3 P/T results

In this section we show the results of experiments involving our structural reductions for P/T nets. We have implemented Rules C and S in `verifypn` [17, 4], that already includes structural reductions Rules A-R which will be used alongside our new rules. The *base* experiment uses the Rule C described in [5], while experiment *base+C* uses our Rule C as defined in Section 5.1. In experiment *base+CS*, we also use our Rule C and the Rule S is appended to the structural-reduction-rule application sequence and is thus used at the end of each iteration

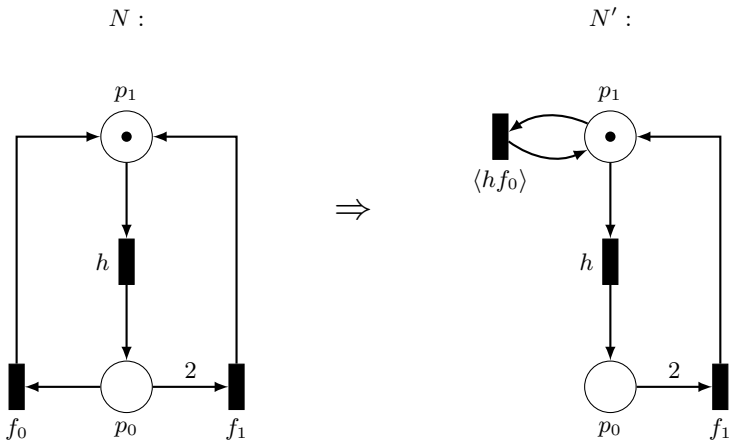


Figure 9: Example of Rule S introducing deadlocks to the P/T net N. In N we have $N \vDash AG\neg deadlock$ while in the reduced net N' after one application of Rule S, we have that $N' \nvDash AG\neg deadlock$ because $M'_0 \xrightarrow{h} \not\rightarrow$.

Experiment	base	base+C	base+CS	
Category \ Rule	C	C	C	S
ReachabilityCardinality	25.4%	25.4%	26.1%	29.8%
ReachabilityFireability	25.8%	25.8%	26.6%	41.3%
LTLCardinality	23.6%	23.6%	23.6%	4.8%
LTLFireability	23.3%	23.3%	23.3%	6.3%
CTLCardinality	22.6%	22.6%	22.7%	1.4%
CTLFireability	20.3%	20.3%	20.3%	1.7%

Table 2: Percentages of queries where the given rule was applied at least once for the given experiment and category

of rule applications. Recall that we are using the P/T nets of MCC2021 for these experiments, and there are 113.376 queries in total, which is 18.896 per category.

In Table 2 we see the percentages of queries where the given rule was used at least once in the given experiment. In Table 3 we see the absolute number of applications of each rule in each experiment across all queries. Although we proved that our Rule C is more general than Rule C by Bønneland et al. [5] in Section 5.1, we find that our Rule C does not result in more applications, and most likely due to noise it is a few times less for ReachabilityCardinality. However, Rule S finds many applications, especially on reachability queries. Despite not being correct for LTL and CTL, Rule S is applied here as well. This is because some LTL and CTL queries in MCC2021 are or can be simplified to a reachability query, which means Rule S can be used anyway. Also recall, that Rule S falls back to a standard pre agglomeration on non-reachability and on deadlock prop-

Experiment	base	base+C	base+CS	
Category ╲ Rule	C	C	C	S
ReachabilityCardinality	228425	228407	233247	7829495
ReachabilityFireability	236375	236375	243677	10874826
LTLCardinality	205686	205686	206740	32307
LTLFireability	222829	222829	241645	56086
CTLCardinality	153761	153761	154210	354542
CTLFireability	126458	126458	126465	901723

Table 3: Number of applications of the given rule in the given experiment and category

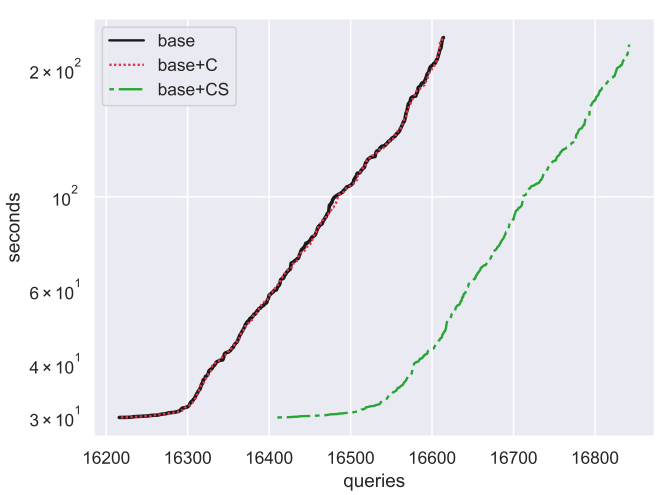
Category ╲ Experiment	base	base+CS	base+CS
ReachabilityCardinality	17482	-3	+144
ReachabilityFireability	16615	-1	+228
LTLCardinality	17488	-1	+9
LTLFireability	17001	-2	+3
CTLCardinality	15217	+1	+5
CTLFireability	12120	+4	+4
Sum	95923	-2	+393

Table 4: The number of answers found by the base experiment and the number of additional answers found with our rules compared to the base experiment.

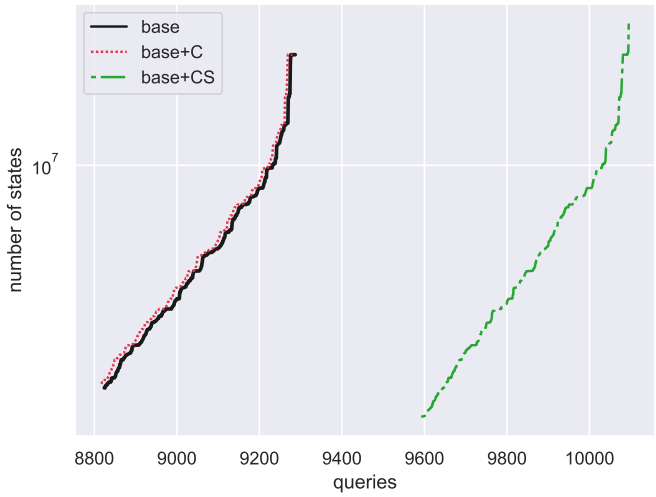
erties, resulting in further applicability on LTL. We can see that when Rule S is used, Rule C are applied more, especially on ReachabilityFireability.

Table 4 shows the number of answers found in each experiment within the time limit of 4 minutes. See Appendix C where we split this table into several tables and show answers gained per model. Our rule C seems to lose a few answers compared to the baseline in the reachability and LTL categories, but gain a few in the CTL. These differences are however not greater than what can be attributed to noise. Although our Rule C is more general than the old Rule C, the benchmark did not contain a significant number of cases where that generality made a difference, which we had hoped. On the other hand, Rule S clearly leads to more results, primarily found in the reachability categories. These new answers are mainly found on the following models: CloudDeployment, DLCflexbar, DLCround, DLCround, DLCshifumi, FlexibleBarrier, MAPK, NoC3x3, and UtilityControlRoom.

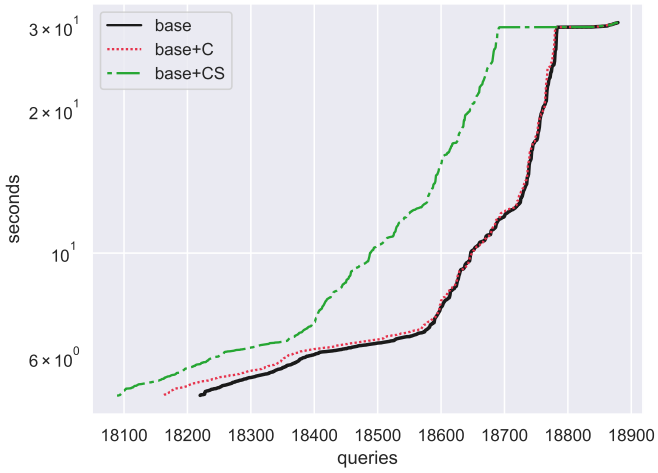
Figure 11 shows the total time spent, time spent in the reduction phase, the time spent in the verification phase, and the state space size of the reduced net for the reachability fireability queries. All four graphs further confirms that Rule C is on par with the base experiment. Rule S reduces the time spent on verification and the size of the state space significantly, and while it does add additional overhead in the reduction phase, the total time is better. We only present the ReachabilityFireability results here, because the improvements of



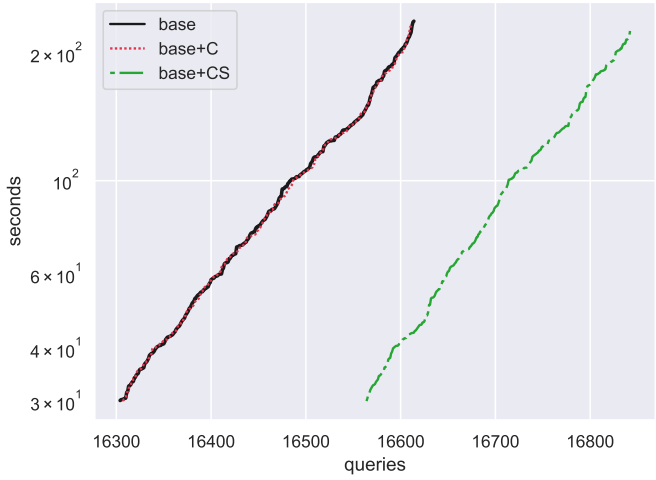
(a) Total time



(b) State space size



(c) Reduce time



(d) Verification time

Figure 11: Reachability fireability test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.

our rules are more modest on the other categories. The graphs for the remaining categories can be found in Appendix C.1. The effects of Rule S are much less pronounced on the other categories, obviously due to the applicability of Rule S on CTL and LTL, while Rule C remains equal to base. The most interesting thing to note is that on LTL fireability Rule C achieves further reduction of the state space size compared to the baseline. However, as discussed earlier, it does not result in more answers.

6 Structural reductions for CPNs

In this section we present our structural reduction rules for CPNs.

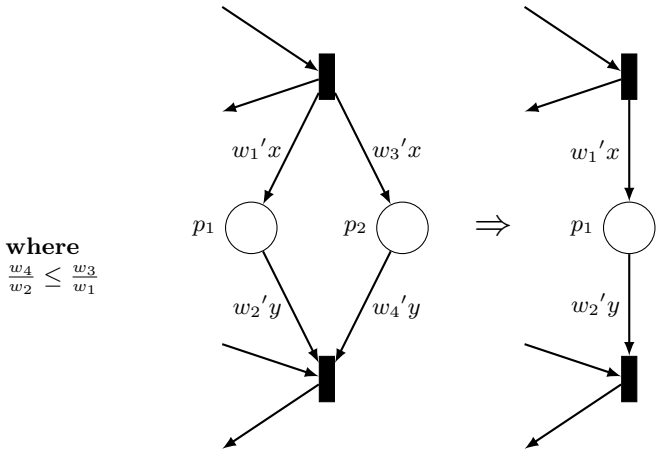
6.1 Rule C: Parallel places (CPN)

We now generalize Rule C from Section 5.1 that removes parallel places to CPNs. Rule C generalized to CPNs is defined in Figure 12. Again, the fraction d describes how quickly tokens can be consumed from p_2 compared to p_1, while f describes how slowly tokens can be fed to p_2 compared to p_1, and if $d \leq f$ then p_2 never disables their post set unless p_1 also does. However, for CPNs we have to take every color into account when defining d and f. Hence, we must compare all tuples of variables that appear on the arcs, but also the supports of the arc multisets and the initial markings must match. Otherwise, it is possible to add a color to one place without adding it to the other, which makes the rule not applicable.

Theorem 6. *The CPN Rule C shown in Figure 12 is correct for CTL*.*

Proof. This proof is similar to the proof for Theorem 4 about the P/T net version of Rule C correctness. We will show that a net reduced by Rule C is bisimilar to the original net using contradiction. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a CPN with initial marking M_0 and let $\varphi \in \text{CTL}^*$. Let N' and M_0' be the CPN and initial marking after applying Rule C once on N. Let M be a reachable marking s.t. $M_0 \to^* M$. To argue for the correctness of Rule C, we will show that if any $t \in T$ is disabled due to a lack of tokens in p_2 in marking M, then it is also disabled due to a lack of tokens in p_1. In other words, that $M(p_2) \subset b(\boxminus(p_2, t))$ implies $M(p_1) \subset b(\boxminus(p_1, t))$ for any $t \in T$ and $b \in \vec{B}(t)$. To show this, we must realize that in conjunction with C5, that d is the fastest rate at which p_2 can be emptied compared to p_1 for any color, and conversely, that f is the slowest rate at which p_2 can be fed compared to p_1 for any color. Both d and f are well defined by convention that $\min \emptyset = -\infty$ and $\max \emptyset = \infty$. Let $D = \mathcal{X}(p_1) = \mathcal{X}(p_2)$ as of C1.

We will now show that $M(p_2) \subset b(\boxminus(p_2, t))$ implies $M(p_1) \subset b(\boxminus(p_1, t))$ by contradiction. Assume that there exists a transition $t \in T$, a binding $b \in \vec{B}(t)$, and a reachable marking M such that $M_0 \to^* M$ where $M(p_1) \not\subset b(\boxminus(p_1, t))$ and $M(p_2) \subset b(\boxminus(p_2, t))$.



Precondition	Update
Fix places p_1 and p_2 s.t.: C1) $\mathcal{X}(p_1) = \mathcal{X}(p_2)$ C2) $p_2 \notin places(\varphi)$ C3) $p_2^\circ = \emptyset$ C4) $p_1^\bullet \neq \emptyset$ C5) For all $t \in T$: $\mathbf{Supp}(\boxminus(p_1,t)) = \mathbf{Supp}(\boxminus(p_2,t)) \wedge$ $\mathbf{Supp}(\boxplus(t,p_1)) = \mathbf{Supp}(\boxplus(t,p_2))$ C6) $\mathbf{Supp}(M_0(p_1)) = \mathbf{Supp}(M_0(p_2)) \wedge$ $M_0(p_1) \cdot d \subseteq M_0(p_2)$ C7) $d \leq f$ where $d = \max_{t \in p_1^\bullet, \vec{v} \in \boxminus(p_1,t)} \frac{\boxminus(p_2,t)(\vec{v})}{\boxminus(p_1,t)(\vec{v})}$ $f = \min_{t \in {}^\bullet p_1, \vec{v} \in \boxplus(t,p_1)} \frac{\boxplus(t,p_2)(\vec{v})}{\boxplus(t,p_1)(\vec{v})}$	UC1) remove p_2

Figure 12: Rule C: Parallel places (CPN)

By C5 and the definition of d, this means that $\exists c \in D.M(p_2)(c)/M(p_1)(c) < d$. By C6, we have that $\forall c \in D.M_0(p_2)(c)/M_0(p_1)(c) \geq d$ and thus $M_0 \neq M$. Since M where $\exists c \in D.M(p_2)(c)/M(p_1)(c) < d$ is reachable there must exist a transition $t' \in T$ and binding $b' \in \vec{B}(t')$ such that firing t' using b' either:

1. consumes at least d times more tokens of color c from p_2 than from p_1, i.e. $b'(\boxminus(p_2,t'))(c)/b'(\boxminus(p_1,t'))(c) > d$, or

2. produces at most d times more tokens of color c to p_1 than to p_2, i.e. $b'(\boxplus(t', p_2))(c)/b'(\boxplus(t', p_1))(c) < d$.

Precondition C5 implies that we cannot add a color to one place without adding at least one to the other place too. Thereby, case 1. is impossible by the definition of d. But case 2. is also impossible due to the C7 and the definition of f. Alas we have a contradiction.

In combination with C3, we can conclude that N' must be bisimilar, and hence Rule C is correct for CTL* by Theorem 1. □

6.2 Rule D: Parallel transitions

Our Rule D extends Rule D by Bønneland et al. from [5] with colors and inhibitor arcs. Rule D handles parallel transitions where the effect of firing one of them is equivalent to firing the other exactly k times. In such a case, we remove the transition with higher arc weights. The definition of Rule D can be seen in Figure 13. In precondition D2, we state that the valid bindings of the guard $G(t_1)$ must be a subset of the valid bindings of $G(t_2)$, i.e. $\vec{B}(t_1) \subseteq \vec{B}(t_2)$. This can be expensive to check depending on the complexity of the guards and the number of variables in the guard. A cheap overapproximation is to check whether $G(t_1) = G(t_2)$ or $G(t_2) = \top$ instead, which we do in our implementation. As seen in D3 we stricten the preconditions of Rule D if the query φ is a CTL property or contains the next operator X. In D7 we loosen the preconditions if φ is a reachability query.

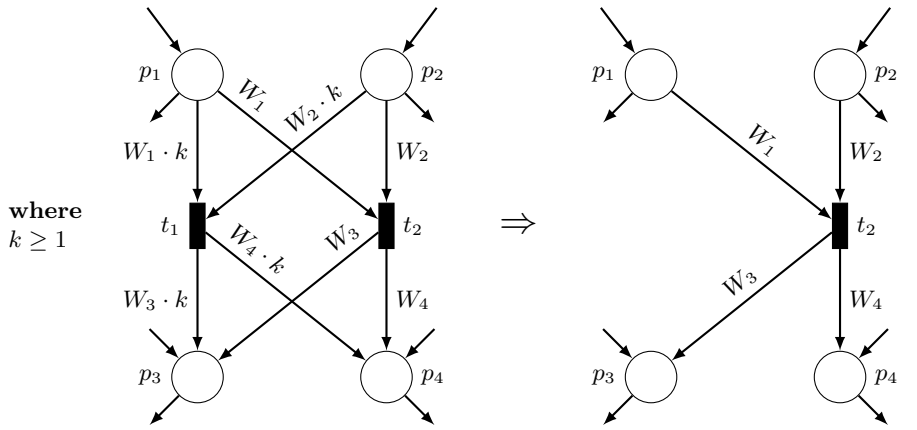
Proving the correctness of Rule D is two-fold. First we prove correctness for LTL\X where $k \geq 1$ and then for CTL* where $k = 1$.

Theorem 7. *Rule D described in Figure 13 is correct for LTL\X.*

Proof. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a CPN with initial marking M_0 and let $\varphi \in$ LTL\X. Let N' and M'_0 be the CPN and initial marking after applying Rule D once on N. Assume $\varphi \notin \mathit{Reach}$. We shall now argue, that Rule D is correct by showing that every maximal path starting from M_0 in N has a stutter-equivalent trace starting from M'_0 in N' and vice versa.

$\Rightarrow$: Let $w \in T^{\bigstar}$ be a maximal trace such that $M_0 \xrightarrow{w_1} M_1 \xrightarrow{w_2} M_2 \xrightarrow{w_2} \ldots$. Let w' be w with all occurrances of t_1 replaces with k times t_2. By D2, we know that if t_1 has a valid binding $b \in \vec{B}(t_1)$, then t_2 has a similar valid binding $b \in \vec{B}(t_2)$. By D4, the effect of firing t_1 with b has the same effect as k firings of t_2 with b. Lastly, by D5 and D6, we have that t_2 is not inhibited unless t_1 is also inhibited and firing t_2 cannot inhibit t_2. Together, the means that any firing of t_1 can be replicated by k firings of t_2. By assumption that $\varphi \notin \mathit{Reach}$, the transitions t_1 and t_2 are invisible to φ. Hence, w' must be stutter-equivalent to w.

$\Leftarrow$: Let $w \in T'^{\bigstar}$ be a maximal trace such that $M'_0 \xrightarrow{w_1} M'_1 \xrightarrow{w_2} M'_2 \xrightarrow{w_2} \ldots$. Since $T' \subseteq T$, the trace w is also trivially fireable in N such that $M_0 \xrightarrow{w_1} M_1 \xrightarrow{w_2} M_2 \xrightarrow{w_2} \ldots$.



Precondition	Update
Fix transitions t_1 and t_2 and $k \in \mathbb{N}$ s.t.: D1) $t_1 \notin transitions(\varphi)$ D2) $\vec{B}(t_1) \subseteq \vec{B}(t_2)$ D3) $\varphi \in \text{CTL} \vee X \in \varphi \implies k = 1$ D4) For all $p \in P$: $\boxminus(p, t_1) = \boxminus(p, t_2) \cdot k$ $\boxplus(t_1, p) = \boxplus(t_2, p) \cdot k$ D5) ${}^\circ t_2 \cap t_2^\bullet = \emptyset$ D6) $\forall p \in P. I(p, t_1) \leq I(p, t_2)$ D7) $\varphi \notin \text{Reach}({}^\bullet t_1 \cup t_1^\bullet) \cap (places(\varphi) \cup {}^\bullet transitions(\varphi)) = \emptyset$	UD1) Remove t_1

Figure 13: Rule D: Parallel transitions

If $\varphi \in \mathit{Reach}$ then every state reachable in N must also reachable in N'. This is easily proven similarly to above, except that the traces do not have to be stutter-equivalent. Hence, by D7 and by Theorem 2 the Rule D is correct for LTL\X. □

Theorem 8. *Rule D described in Figure 13 is correct for CTL* if $k = 1$.*

Proof. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a CPN with initial marking M_0 and let $\varphi \in$ CTL*. Let N' and M_0' be the CPN and initial marking after applying Rule D once on N with $k = 1$. We shall now argue, that Rule D is correct.

Let $\equiv_D \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ be an equivalence relation such that $M \equiv_D M'$ iff $M = M'$. Since Rule D does not remove places, obviously $M_0 \equiv_D M_0'$.

We will now show that if $M \equiv_D M'$ then:

1. M and M' satisfies the same propositions w.r.t. φ,

2. if $M \xrightarrow{t} M_1$ then either $M' \xrightarrow{t} M'_1$ or $M' \xrightarrow{t_2} M'_1$ s.t. $M_1 \equiv_D M'_1$, and
3. if $M' \xrightarrow{t} M'_1$ then $M \xrightarrow{t} M_1$ s.t. $M_1 \equiv_D M'_1$.

1. : Let $M \equiv_D M'$. Since Rule D only removes t_1. By 1. and 2. the deadlock proposition is preserved. So together with D1, M and M' satisfies the same propositions w.r.t. φ.

2. : Let $M \equiv_D M'$ and let $M \xrightarrow{t} M_1$. There are two cases:

- Case $t \neq t_1$. Trivially, $M' \xrightarrow{t} M'_1$ and $M_1 \equiv_D M'_1$.
- Case $t = t_1$. By D6 we have that t_2 is not inhibited, otherwise t_1 would have been inhibited too. By precondition D2 and D4 we know that the effect of firing t_1 can be replicated by firing t_2 exactly k times using a similar valid binding, and $k = 1$. Therefore, it must be the case that $M' \xrightarrow{t_2} M'_1$ and $M_1 \equiv_D M'_1$.

3. : Let $M \equiv_D M'$ and let $M' \xrightarrow{t} M'_1$. Trivially, $M \xrightarrow{t} M_1$ and $M_1 \equiv_D M'_1$.

In conclusion $\equiv_D$ is a bisimulation, and hence by Theorem 1, Rule D must be correct for CTL* when $k = 1$. □

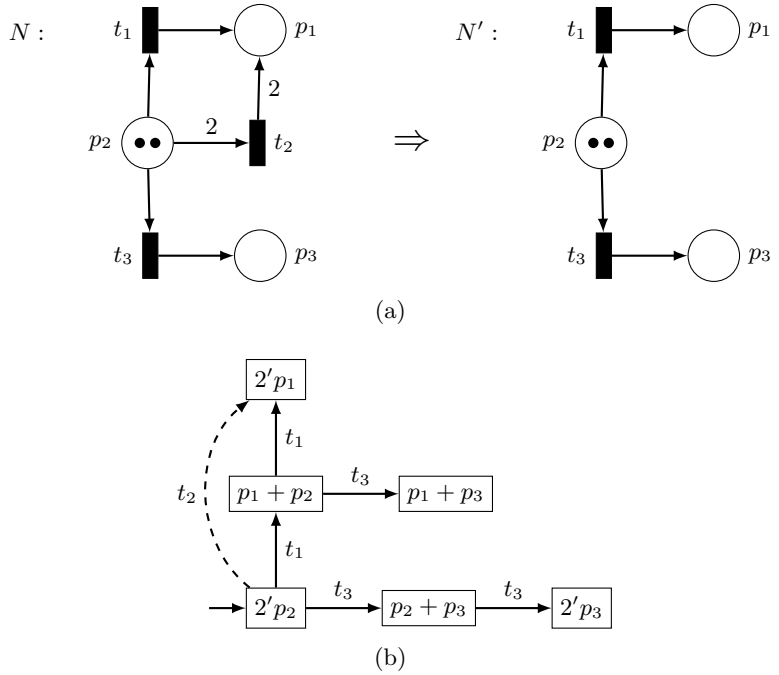
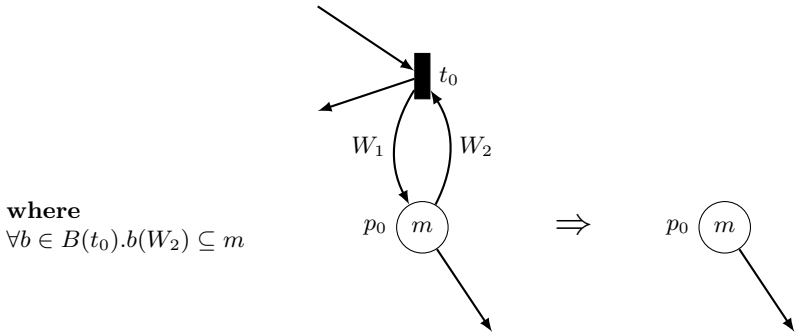


Figure 14: Example of Rule D not preserving a CTL\X formula $\varphi = E[(EFp_3 = 2)U(\neg EFp_3 > 0)]$. (a) shows two P/T nets N and N' where N' is the net N after one application of Rule D (with $k = 2$). We have that $N \vDash \varphi$ while $N' \nvDash \varphi$. (b) shows the underlying transitions system of N and N'. Transtion t_2 only exist in N.

Obviously, Rule D does not preserve stutter-sensitive properties unless $k = 1$. The reason that $k = 1$ for CTL\X in D3 is shown in Figure 14. We have that $\varphi = E[(EFp_3 = 2)U(\neg EFp_3 > 0)] \in$ CTL\X and $N \vDash \varphi$ while $N' \nvDash \varphi$. This is easily seen in Figure 14b, which shows the underlying transition system of N and N'. In N, we have $2'p_2 \vDash EFp_3 = 2$ and if t_2 is fired, then the resulting state is $2'p_1$ and $2'p_1 \vDash \neg EFp_3 > 0$, and hence φ is satisfied. It is not possible to fire t_2 in N' and there exists no path where $EFp_3 = 2$ is satisfied until $\neg EFp_3 > 0$ is satisfied.

6.3 Rule E: Redundant transitions

We can remove transitions that will never be enabled from the initial marking, without removing behaviour from the net. The exact definition of this reduction is shown in Figure 15. We make sure that the initial marking of the given place does not enable the given consumer, and that the place will not receive more tokens in the future. We also make sure that the transition is never enabled, by checking all bindings in the current marking for the place p_0, seeing if any available binding enables the transition t_0. If there are too many bindings, we skip this check and do not consider removing the transition, due to the overhead in applying the rule. A further improvement in this check, is to do a simple overapproximation, where we only consider the cardinalities of the marking $M_0(p_0)$ and $\boxminus(p_0, t_0)$. As we have that if $|M_0(p_0)| < |\boxminus(p_0, t_0)|$, no binding makes firing of



Precondition	Update
Fix place p_0 and transition t_0 s.t.: E1) $\forall b \in B(t).b(\boxminus(p_0, t_0)) \not\subseteq M_0(p_0)$ E2) $\forall t \in {}^\bullet p_0 : \boxplus(t, p_0) \subseteq \boxminus(p_0, t)$ or $\forall b \in B(t).b(\boxminus(p_0, t)) \not\subseteq M_0(p_0)$ E3) $t_0 \notin transitions(\varphi)$	UE1) If $p_0 \notin places(\varphi)$ and $p_0^\bullet = \{t_0\}$, remove p_0 UE2) Remove t_0

Figure 15: Rule E: Redundant transitions

this transition possible. We do this cardinality check before evaluating bindings, as this cardinality check is faster, and if this fails, the binding check will also fail. If these conditions are met, we can safely remove the consumer. If this leaves the place with no consumers, and the place is not mentioned in the query, we can remove the place as well. This rule is an extension of the Rule E from [5], but adapted to work on CPNs.

Precondition E3 can be ignored if we allow rewriting of φ, in which case all instances of $en(t_0)$ can be replaced with $\neg\top$.

Theorem 9. *Rule E in Figure 15 is correct for CTL* queries.*

Proof. By precondition E1 transition t_0 is not enabled in $M(p_0)$, no matter the binding of the variables on the arcs. Furthermore, E2 tells us that all consumers of p_0 are either also disabled in marking M_0, or that the effect of firing it is to remove tokens from p_0. This means that no firing of any $t \in T$, will add tokens to p_0 and thereby enable t_0. Therefore, we can safely remove t_0 as this transition will never be enabled. □

6.4 Rule F: Redundant places

The idea behind Rule F, shown in Figure 16, is to remove places, which will always have enough tokens with the correct colors to fire its consumers, making it redundant. For this rule, it makes sense to check every binding of the consumers, since an overapproximation may make the rule applicable in very few cases.

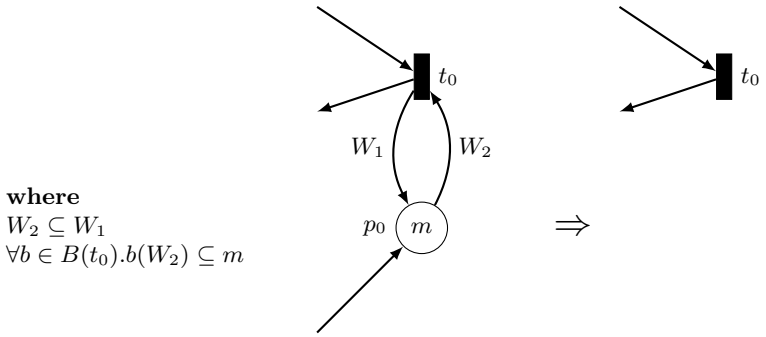
However, if there are too many bindings, the overhead in applying the rule will eliminate the benefit. Therefore, if there are more than 10000 bindings to check for precondition F4, we skip the application of this rule on the given place. Our Rule F is a colored extension of Rule F presented by Bønneland et al. in [5].

Theorem 10. *Rule F in Figure 16 is correct for any CTL* formula.*

Proof. By precondition F3 we know that all transition $t \in p_0^\bullet$ can only add tokens to p_0. Additionally, by F4 we know that t can fire using all bindings $B(t)$ in the initial marking M_0. As a result, p_0 can never disable any transition $t \in p_0^\bullet$. It will also never inhibit by F2. Combined with F1 is therefore safe to remove p_0 as it does not change the behaviour of the net. □

6.5 Rule I: Irrelevant places and transitions

Only some places and transitions are relevant for the query. Algorithm 1 shows how we can remove everything that is irrelevant. Overall the algorithm is straight forward. We start by marking the transitions that we know are relevant for the query, i.e the transitions of $transitions(\varphi)$ and ${}^\bullet places(\varphi) \cup places(\varphi)^\bullet$. Then we traverse the net to find and mark all transitions that can enable any of these. Lastly, we remove all transitions that was not marked, and remove all places not in the preset of any marked transition. This algorithm is very similar to Rule I presented by Bønneland et al. in [5] which works for P/T nets. However, for



Precondition	Update
Fix place p_0 s.t.: F1) $p_0^\circ = \emptyset$ F2) $p_0 \notin places(\varphi)$ and for all $t \in p_0^\bullet$: F3) $\boxminus(p_0, t) \subseteq \boxplus(t, p_0)$ F4) $\forall b \in B(t).b(\boxminus(p_0, t)) \subseteq M_0(p_0)$	UF1) remove p_0

Figure 16: Rule F: Redundant places

CPNs there is another way to enable a transition, and that is by transmuting the color of tokens. Hence, in line 7, we have to not only consider transitions with positive effect on the preset of t, i.e. transitions that increase the number of tokens, but also transitions that modify the colors of tokens in the preset of transition t. In other words we enqueue ${}^{\nabla}({}^\bullet t)$ instead of ${}^{\boxplus}({}^\bullet t)$, where

$$ {}^{\nabla}p = \{t \in {}^\bullet p \mid \boxplus(t, p) \neq \boxminus(p, t)\} $$

is the transmuting preset of $p \in P$ af defined in Section 3.

Theorem 11. *Rule I in Algorithm 1 is correct for reachability without deadlock.*

Proof. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G\rangle$ be a CPN with initial marking M_0 and let $\varphi \in \mathrm{EF}\backslash X, deadlock$. Let N' and M_0' be the CPN and initial marking after applying Rule I (Algorithm 1) once on N. Let P' and T' be the places and transitions of the net N', respectively. We shall now argue that Rule I is correct using the relation $\equiv_I \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ where $M \equiv_I M'$ if and only if

1. $M(p) = M'(p)$ for all $p \in places(\varphi)$,
2. $M(p) \subseteq M'(p)$ for all $p \in {}^\bullet T'$, and
3. $|M(p)| \geq |M'(p)|$ for all $p \in {}^\circ T'$.

Algorithm 1: Rule I: Irrelevant places and transitions

Input: A CPN $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$, initial marking M_0 and EF formula φ without *deadlock*

Output: A reduced net N' and its initial marking M_0'

```
1  X := ∅                                                  /* Relevant transitions */
2  Q := transitions(φ) ∪ •places(φ) ∪ places(φ)•           /* Queue of transitions */
3  while Q ≠ ∅ do
4  |   Pick any t ∈ Q
5  |   Q := Q \ {t}
6  |   X := X ∪ {t}                                       /* Mark as relevant */
7  |   Q := Q ∪ ▽(•t) \ X      /* Enqueue transitions that can enable t */
8  |   Q := Q ∪ (°t)⊟ \ X
9  P' := •X ∪ °X ∪ places(φ)
10 T' := X
11 N' := a copy of N but every place p ∉ P' and every transition t ∉ T' have
       been removed.
12 M0' := a marking s.t. M0'(p) = M0(p) for all p ∈ P'.
13 return N' and M0'
```

Let $M \equiv_I M'$. Then clearly, $M \vDash \varphi$ iff $M' \vDash \varphi$ due to the definition of $\equiv_I$, and the fact that $places(\varphi) \subseteq P'$ and $transitions(\varphi) \subseteq T'$. Moreover, we have that $M_0 \equiv_I M_0'$.

$\Rightarrow$: Let $M \equiv_I M'$. We will now show that if $M \xrightarrow{t,b} M_1$ then either $M' \xrightarrow{t,b} M_1'$ such that $M_1 \equiv_I M_1'$ or $M_1 \equiv_I M'$. There are two cases:

- Case $t \in T'$. Due to the second and third condition in the definition of $\equiv_I$, we have that t is also enabled in M' using binding b and so $M' \xrightarrow{t,b} M_1'$. By definition, firing t using binding b preserves all three conditions of $\equiv_I$ and hence $M_1 \equiv M_1'$.
- Case $t \notin T'$. We shall argue that $M_1 \equiv_I M'$. Algorithm 1 ensures that ${}^{\triangledown}({}^{\bullet}T') \subseteq T'$ and $({}^{\circ}T)^{\boxminus} \subseteq T'$. Hence, firing $t \notin T'$ in N cannot add or transmute tokens in any place from ${}^{\bullet}T'$ and cannot remove tokens from any place in ${}^{\circ}T'$. Additionally, the firing of t cannot change the tokens in $places(\varphi)$ as ${}^{\bullet}p \cup p^{\bullet} \subset T'$ for all $p \in places(\varphi)$. As a result, the three conditions in the definition of $\equiv_I$ are preserved and $M_1 \equiv_I M'$.

$\Leftarrow$: For this direction, it is enough to notice that N' is a subnet of N. Hence, whenever $M_0' \xrightarrow{w} M'$ for some $w \in T'^*$ then also $M_0 \xrightarrow{w} M$ and $M \equiv_I M'$.

□

Rule I does not preserve deadlock- and stutter sensitive properties, because the removed parts can contain loops.

6.6 Rules T and U: Pre agglomeration and atomic free agglomeration

This section covers our colored variants of pre agglomerations and atomic free agglomerations. While colored pre agglomerations have already been presented by Evangelista et al. in [12], we present Rule T, an alternative definition of colored pre agglomerations matching the other definitions of structural reduction rules in TAPAAL as by in [5]. The atomic free agglomeration for P/T nets, which we present as Rule S in Section 5.2, has a colored variant which can be found in Appendix B. However, we found that it performs better when restricted to $k = 1$, and therefore we instead present Rule U in this section, which is equivalent to Rule S for CPNs, but with $k = 1$.

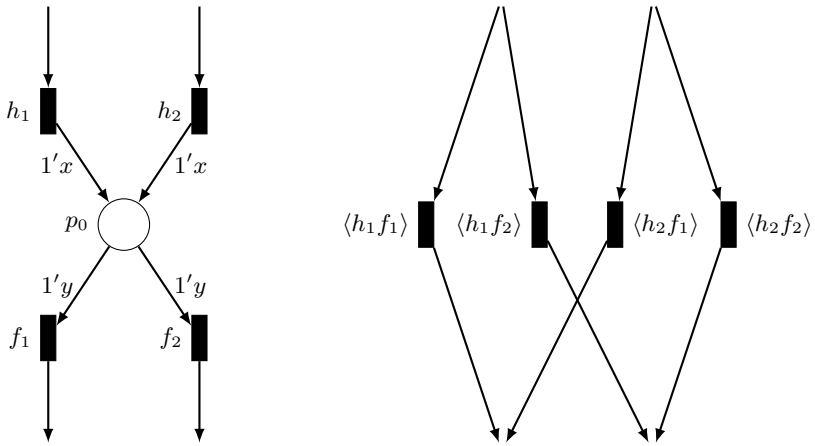


Figure 17: A example of an agglomeration in a CPN

Figure 17 shows an example of a simple colored agglomeration. The basis of a pre agglomeration around a place p_0, is establishing that there is no meaningful difference between firing a producer $h \in {}^\bullet p_0$, and postponing that firing until the moment a consumer $f \in p_0^\bullet$ needs the token(s) produced by h. When this is the case for all pairs of producers and consumers, the place p_0 can be replaced entirely by a series of transitions $\langle hf \rangle$ representing each such combination of firing a producer and consumer.

The atomic free agglomeration agglomerates individual consumers of the place p_0. In cases where all consumers $f \in p_0^\bullet$ can be agglomerated, this replicates the outcome of a pre agglomeration. But if not all f fulfill the requirements for agglomeration, the atomic free agglomeration can still be applied and replace individual consumers $f_0 \in p_0^\bullet$ that do fulfill the conditions. In this case the place p_0, the producers ${}^\bullet p_0$, and $f \in p_0^\bullet \setminus \{f_0\}$ will not be removed from the net. Ap-

plying this rule will then increase the overall number of transitions in the net, but reduce the interleavings in the underlying transition system. This can also introduce deadlocks where h is fired and the tokens get stuck in p_0, exactly as shown in Figure 9.

The main differences between agglomerations on CPNs and P/T nets arise from the fact that tokens are not interchangeable. This means that a consumer $f \in p_0^\bullet$ can have different behavior depending on the color of the token it consumes from p_0. A firing of f that consumes tokens produced by two separate producers $h_1, h_2 \in {}^\bullet p_0$ is no longer necessarily interchangeable with a firing consuming tokens from only h_1. This also extends to separate firings of the same producer, with different bindings, such as h_1 being fired twice with different bindings for x. As such, the preconditions for the agglomerations are modified to only apply in cases where the replacement transitions $\langle hf \rangle$ represent not only each firing sequence hf, but exactly the same possible variable bindings of the original firing sequence.

This is accomplished by restricting the arc expressions on the arcs in and out of p_0 to only contain exactly one color at a time. Under that condition, there is exactly one way to match any binding of variable x to a binding of variable y, and any firing of the consumer can be attributed to exactly one firing of a producer that enabled it. The other restriction on bindings is guards. As each replacement transition $\langle hf \rangle$ simply represents firings of h and f in sequence, the guards restricting either one of them also restrict the firing sequence, so the guard of $\langle hf \rangle$ is simply a conjunction of the guards of h and f. This means the guard of f now also prevents the firing of h, but as mentioned earlier in this section, the basis of the pre- and free agglomerations is that delaying h until f is ready to fire is valid.

For the following formal definitions, we define a transformation function for the renaming of variables:

Definition 9. *The transformation $rename(t, v, v')$ replaces all occurrences of variable v with variable v' in the guard and in all arcs of transition t.*

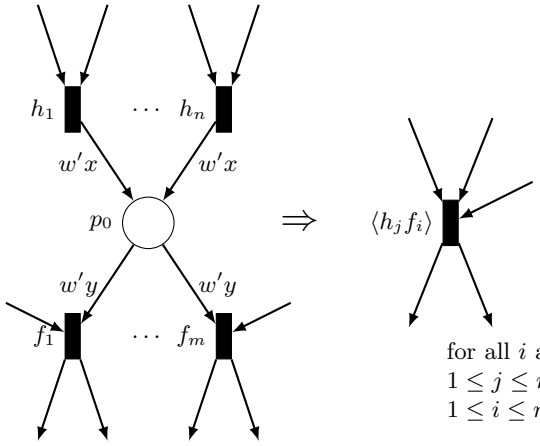
The definition of the pre agglomeration, Rule T, can be found in Figure 18. The definition of the atomic free agglomeration, Rule U, can be found in Figure 19.

Theorem 12 ([12]). *Rule T in Figure 18 is correct for $LTL\backslash X$.*

Theorem 13. *Rule U in Figure 19 is correct for reachability without deadlock.*

Proof. This proof will follow the same approach as the proof for Rule S for P/T nets in Section 5.2. It will not include the considerations for k-scaled arc weights as these are not allowed by U10. It will include how the given preconditions are sufficient for the proof method to hold for CPNs.

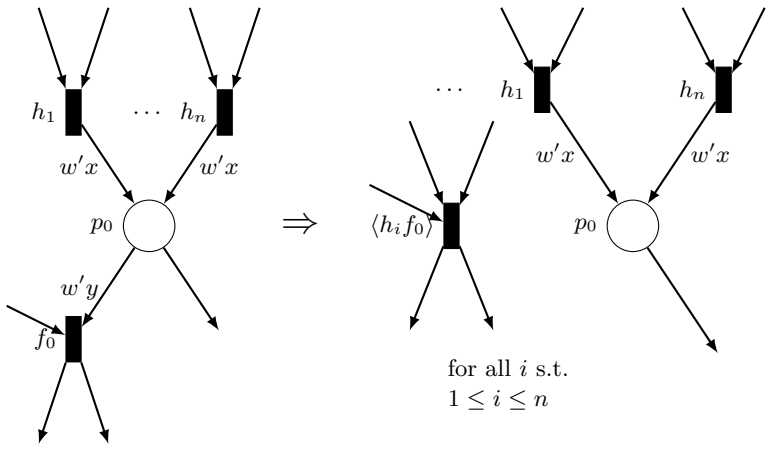
Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G \rangle$ be a CPN with initial marking M_0 and let $\varphi \in$ Reach without deadlock. Let N' and M_0' be the CPN and initial marking after applying Rule U once on N.



Precondition	Update
Fix place p_0 s.t.: T1) $(\{p_0\} \cap places(\varphi) = \emptyset$ T2) $(p_0^\bullet \cup {}^\bullet p_0) \cap transitions(\varphi) = \emptyset$ T3) $M_0(p_0) = \emptyset$ T4) ${}^\bullet p_0 \cap p_0^\bullet = \emptyset$ and for all $h \in {}^\bullet p_0$: T5) $({}^\bullet h)^\bullet = \{h\}$ T6) $h^\bullet = \{p_0\}$ T7) ${}^\bullet h \cap places(\varphi) = \emptyset$ T8) $p_0^\circ = {}^\circ h = ({}^\bullet h)^\circ = \emptyset$ and for all $f \in p_0^\bullet$ T9) $\lvert\mathbf{Supp}(\boxplus(h, p_0))\rvert = \lvert\mathbf{Supp}(\boxminus(p_0, f))\rvert = 1$ T10) $\lvert \boxplus (h, p_0)\rvert = \lvert \boxminus (p_0, f)\rvert$	For all $h \in {}^\bullet p$, for all $f \in p^\bullet$, create a transition $\langle hf \rangle$ s.t. for all $p \in P \setminus \{p_0\}$: UT1) For all $v \in \mathbf{Vars}(f)$, $rename(f, v, v')$ with some $v' \in \mathbf{Vars}_{\mathcal{X}(p)} \backslash \mathbf{Vars}(h)$ then UT2) $\boxminus(p, \langle hf \rangle) := \boxminus(p, h) \uplus \boxminus(p, f)$ UT3) $\boxplus(\langle hf \rangle, p) := \boxplus(f, p)$ UT4) $G(\langle hf \rangle) := G(h) \wedge G(f)$ UT5) $I(\langle hf \rangle) := I(f)$ then UT6) Given that $\boxplus(h, p_0) = w'\langle x_1, x_2, \ldots, x_n \rangle$ and $\boxminus(p_0, f_0) = w'\langle y_1, y_2, \ldots, y_n \rangle$ For $i \in [1, n]$ Let a be the smallest index s.t. $x_a = x_i$ holds: $rename(\langle hf \rangle, x_i, y_a)$ $rename(\langle hf \rangle, y_i, y_a)$ then after all such transitions are made: UT7) Remove $p^\bullet$, ${}^\bullet p_0$, and p_0

Figure 18: Rule T: Pre agglomeration

Let $w \in T^*$ be a trace in N such that $M_0 \xrightarrow{w} M$ for some marking M. We shall now show how w can be transformed into a trace $w' \in T'^*$ such that $M_0' \xrightarrow{w'} M'$ in N' and where M and M' are equivalent w.r.t. the atomic propositions of φ. First, we transform w to w'', and then we transform w'' to w'.



Precondition	Update
Fix place p_0 and transition f_0 s.t.: U1) $(\{p_0\} \cap places(\varphi) = \emptyset$ U2) $({}^\bullet p_0 \cup p_0^\bullet \cup ({}^\bullet({}^\bullet p_0)^\bullet)) \cap transitions(\varphi) = \emptyset$ U3) $M_0(p_0) = \emptyset$ U4) ${}^\bullet p_0 \cap p_0^\bullet = \emptyset$ U5) $f_0 \in p_0^\bullet$ and for all $h \in {}^\bullet p$: U6) $\lvert\mathbf{Supp}(\boxplus(h, p_0))\rvert = \lvert\mathbf{Supp}(\boxminus(p_0, f_0))\rvert = 1$ U7) $h^\bullet = \{p_0\}$ U8) ${}^\bullet h \cap places(\varphi) = \emptyset$ U9) $p_0^\circ = {}^\circ h = ({}^\bullet h)^\circ = \emptyset$ U10) $\lvert \boxplus (h, p_0)\rvert = \lvert \boxminus (p_0, f_0)\rvert$	For all $h \in {}^\bullet p_0$, create a transition $\langle h f_0 \rangle$ s.t. for all $p \in P \setminus \{p_0\}$: UU1) For all $v \in \mathbf{Vars}(f_0)$, $rename(f_0, v, v')$ with some $v' \in \mathbf{Var}_{\mathcal{X}(p)} \setminus \mathbf{Vars}(h)$ then UU2) $\boxminus(p, \langle h f_0 \rangle) := \boxminus(p, h) \uplus \boxminus(p, f_0)$ UU3) $\boxplus(\langle h f_0 \rangle, p) := \boxplus(f_0, p)$ UU4) $G(\langle h f_0 \rangle) := G(h) \wedge G(f_0)$ UU5) $I(\langle h f_0 \rangle) := I(f_0)$ then UU6) Given that $\boxplus(h, p_0) = w' \langle x_1, x_2, \ldots, x_n \rangle$ and $\boxminus(p_0, f_0) = w' \langle y_1, y_2, \ldots, y_n \rangle$ For $i \in [1, n]$ Let a be the smallest index s.t. $x_a = x_i$ holds: $rename(\langle h f_0 \rangle, x_i, y_a)$ $rename(\langle h f_0 \rangle, y_i, y_a)$ then after all such transitions are made: UU7) Remove f_0 UU8) If $p_0^\bullet = \emptyset$, remove p_0 and all transitions in ${}^\bullet p_0 \setminus transitions(\varphi)$

Figure 19: Rule U: Pre agglomeration

By U4 and U7, h only produces tokens to p_0, and by U1, U2, U8, and U9, h does not affect which transitions are inhibited or the atomic propositions of φ.

This implies that a firing of $h \in {}^\bullet p_0$ can be delayed until a firing of $f \in p_0^\bullet$ needs the tokens that the firing of h produces.

By U10, all arcs from $h \in {}^\bullet p_0$ to p_0 and the arc from p_0 to f_0 has the same weight, and by U6, the arcs produces or consumes exactly one color. Hence, a firing of any h can feed exactly one firing of f_0 if the guards allow it. By U3, p_0 contains no tokens in M_0, and thus every firing of f_0 in w can be matched with a previous firing of an $h \in {}^\bullet p_0$ which produced the tokens which the firing of f_0 consumes.

Let $w'' \in T^*$ be w but where the firings of $h \in {}^\bullet p_0$ which later feed a firing of f_0 are delayed to right before that firing of f_0. By the observations above, we have that $M_0' \xrightarrow{w''} M$ too.

Rule U creates for each $h \in {}^\bullet p_0$ a transition $\langle hf_0 \rangle$. We shall now argue that firing $\langle hf_0 \rangle$ is equivalent to firing the sequence hf_0. Initially, UU1 renames variables which h and f_0 have in common in order to avoid name clashing of variables. Then UU2 and UU3 ensures that $\langle hf_0 \rangle$ have the same effect on $p \in P \setminus \{p_0\}$ as the sequence hf_0. By U6 we have that h and f_0 produces and consumes a single color, respectively. UU6 ensures that this is the same color, by renaming the corresponding variables. Hence, by UU4, UU5, and UU6, the resulting transition $\langle hf_0 \rangle$ is fireable if and only if hf_0 is fireable, and there exists a valid binding $b \in \vec{B}(\langle hf_0 \rangle)$ such that firing $\langle hf_0 \rangle$ using b have the same effect as the sequence hf_0 using any pair of valid bindings. The transition $\langle hf_0 \rangle$ is therefore equivalent to the sequence hf_0, and by U1 and U9, both h, f_0, and $\langle hf_0 \rangle$ are invisible to φ. Let $w' \in T'$ be w'' but where every sequence of hf_0 is replaced with $\langle hf_0 \rangle$. We have that $M_0' \xrightarrow{w'} M'$ s.t. M and M' is equivalent to M w.r.t. the atomic propositions of φ.

Finally, we will show the converse: That if $w' \in T'^*$ is a trace in N' such that $M_0' \xrightarrow{w'} M'$ for some marking M', then w' can be transformed into a trace $w \in T^*$ such that $M_0 \xrightarrow{w} M$ in N and where M and M' are equivalent w.r.t. the atomic propositions of φ. As argue above, there exist valid bindings such that the effect of $\langle hf_0 \rangle$ is equivalent to the effect of the sequence hf_0. Hence w is w' but where every occurrence of $\langle hf_0 \rangle$ is replace with the sequence hf_0. □

Both Rule T and Rule U have an edge case where guards referring to variables that only appear in the arcs to or from p_0 after the reduction can end up referring to variables that do not appear on the transition's guards. While technically valid, these guards can vastly expand the possible number of bindings. As such it may be worth making a function for normalizing such guards where possible, or disallowing the conditions under which they are made. Our implementation does the latter, by including an extra precondition that explicitly disallows the rules if the variable matching used in UT6/UU6 would result in $Vars(G(\langle hf \rangle))$ containing any variables that do not appear on any arc of $\langle hf \rangle$.

A more thorough textual description of the purpose of each precondition can be found in Appendix B.1. Also covered in Appendix B is a description of the variation of atomic free agglomeration for CPN that we experimented with but

did not cover in the main thesis because of its similarity to and worse test results than Rule U.

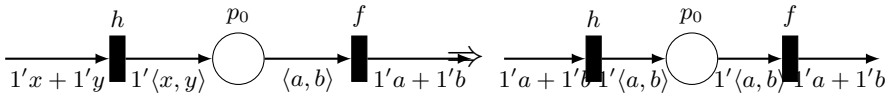


Figure 20: If h is always immediately followed by f and p_0 has an empty initial marking, $\langle x, y\rangle = \langle a, b\rangle$ will always hold as p_0 never retains any tokens outside of hf sequences.

6.7 Rule Q: Preemptive firing

Rule Q, defined in Figure 21, does not reduce structures in the net as the other rules, but will instead move tokens by simulating firing of transitions, which reduces the state space size and sometimes allow other rules to remove the emptied places. The fired transition t_0 must be the only consumer of its pre set. Furthermore, its guard must be $G(t_0) = \top$ and every in- and out going arc must have the same variables. This way, every token is essentially just transferred through the transition. Of course, transition t_0, its pre set, post set, and the post set of its post set, cannot be in the query.

Remark 2. If t_0 is part of a loop in the net, then it is possible for Rule Q to be applied indefinitely. We avoid this by applying Rule Q at most once on each transition.

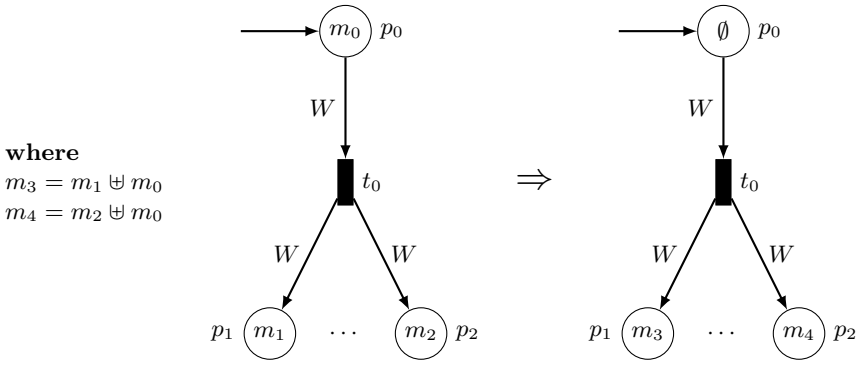
Theorem 14. *Rule Q in Figure 21 is correct for CTL*\X.*

Proof. Let $N = \langle P, T, \mathcal{X}, \boxminus, \boxplus, I, G\rangle$ be a CPN with initial marking M_0 and let $\varphi \in$ CTL*\X. Let N' and M_0' be the CPN and initial marking after applying Rule Q once on N. We argue that Rule Q is correct using an equivalence relation $\equiv_Q \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ such that $M \equiv_Q M'$ iff:

- $M(p) = M'(p)$ for all $p \in P \setminus (\{p_0\} \cup t_0^\bullet)$, and
- there exists a multiset $m \subseteq M_0(p_0)$ such that $n \cdot |\boxminus(p_0, t_0)| = |m|$ for some $n \in \mathbb{N}_0$ and $M(p_0) \setminus m = M'(p_0)$ and $M(p) \uplus m = M'(p)$ for all $p \in t_0^\bullet$.

Clearly, by UQ1 and UQ2, $M_0 \equiv_Q M_0'$. Our proof now follows from the next three properties. Let $M \equiv_Q M'$, then

1. M and M' satisfies the same propositions w.r.t. φ,
2. if $M \xrightarrow{t} M_1$ then either $M_1 \equiv_Q M'$ or $M' \xrightarrow{t} M_1'$ s.t. $M_1 \equiv_Q M_1'$, and



Precondition	Update
Fix place p_0 and transition t_0 s.t.: Q1) $p_0^\bullet = \{t_0\}$ and ${}^\bullet t_0 = \{p_0\}$ Q2) $G(t_0) = \top$ Q3) $(\{p_0\} \cup t^\bullet) \cap places(\varphi) = \emptyset$ and $(\{t_0\} \cup (t^\bullet)^\bullet) \cap transitions(\varphi) = \emptyset$ Q4) $p_0^\circ = \emptyset$ and $(t_0^\bullet)^\circ = \emptyset$ Q5) ${}^\circ t_0 = \emptyset$ Q6) $\exists k \in \mathbb{N}.k \cdot \lvert \boxminus (p_0, t_0) \rvert = \lvert M_0(p_0) \rvert$ Q7) ${}^\bullet p_0 \neq \emptyset \implies \lvert \boxminus (p_0, t_0) \rvert = 1$ and for all $p \in t_0^\bullet$: Q8) $\mathcal{X}(p) = \mathcal{X}(p_0)$ Q9) $\boxminus(p_0, t_0) = \boxplus(t_0, p)$	UQ1) $\forall p \in t_0^\bullet . M_0'(p) \coloneqq M_0(p) \uplus M_0(p_0)$ UQ2) $M_0'(p_0) \coloneqq \emptyset$

Figure 21: Rule Q: Preemptive firing

3. if $M' \xrightarrow{t} M_1'$ then $M^{(i)}$ is M after i firings of t_0 and $M^{(0)} \xrightarrow{t_0} M^{(1)} \xrightarrow{t_0} \cdots \xrightarrow{t_0} M^{(n)} \xrightarrow{t} M_1$ s.t. $M^{(i)} \equiv_Q M'$ for $0 \leq i \leq n$ and $M_1 \equiv_Q M_1'$.

1. : Let $M \equiv_Q M'$. We have that M and M' only differs in the places of $\{p_0\} \cup t_0^\bullet$, so by Q1 and Q3, we know that M and M' satisfies the same propositions w.r.t. φ.

2. : Let $M \equiv_Q M'$ and let $M \xrightarrow{t} M_1$. By definition of $\equiv_Q$ there exists a multiset $m \subseteq M_0(p_0)$ such that $n \cdot \lvert \boxminus (p_0, t_0) \rvert = \lvert m \rvert$ for some $n \in \mathbb{N}_0$ and $M(p_0) \setminus m = M'(p_0)$ and $M(p) \uplus m = M'(p)$ for all $p \in t_0^\bullet$. There are three cases:

- Case $m = \emptyset$, i.e. $M(p) = M'(p)$ for all $p \in P$: Trivially, since Rule Q does not alter the net except the initial marking, we have that $M' \xrightarrow{t} M_1'$ s.t. $M_1 = M_1'$ and thus $M_1 \equiv_Q M_1'$.
- Case $m \neq \emptyset$ and $t = t_0$. We will show that $M_1 \equiv_Q M'$ by showing that there exists a multiset $m_1 \subseteq M_0(p_0)$ and an integer n_1 such that $n_1 \cdot \lvert \boxminus (p_0, t_0) \rvert =$

$|m_1|$ and $M_1(p_0) \setminus m_1 = M'(p_0)$ and $M_1(p) \uplus m_1 = M'(p)$ for all $p \in t_0^\bullet$. If ${}^\bullet p_0 \neq \emptyset$, then by Q7, $W = \boxminus(p_0, t_0)$ contains exactly one variable, $|W| = 1$, which implies that $n_1 = |m_1|$. By Q2, Q5, and Q9, it follows that there exists a binding $b \in \vec{B}(t_0)$ such that $m_1 = m \setminus b(W)$ and $M_1(p_0) \setminus m_1 = M'(p_0)$ and $M_1(p) \uplus m_1 = M'(p)$ for all $p \in t_0^\bullet$. And since t_0 only affects $\{p_0\} \cup t_0^\bullet$, it follows that $M_1 \equiv_Q M'$. Otherwise, if ${}^\bullet p_0 = \emptyset$, then W may contain more variables as of Q7, so $|W| \geq 1$ and we must now prove that $n_1 \cdot |W| = |m_1|$. However, since ${}^\bullet p_0 = \emptyset$, the only tokens to ever occur in p_0 in N are those from the initial marking $M_0(p_0)$ or a subset thereof. Specifically, we have that $M(p_0) = m$, because by Q1, t_0 is the only transition removing tokens from p_0. By Q2, Q5, and Q9, this implies that there exists a binding $b \in \vec{B}(t_0)$ such that $m_1 = m \setminus b(W)$ and $n_1 = n - 1$ and that $n_1 \cdot |W| = |m_1|$. So again, we find that $M_1 \equiv_Q M'$.

– Case $m \neq \emptyset$ and $t \neq t_0$. We will show that $M' \xrightarrow{t} M_1'$ such that $M_1 \equiv_Q M_1'$. Together, Q1 and the definition of $\equiv_Q$, implies that $M(p) \subseteq M'(p)$ for all $p \in {}^\bullet t$. Additionally, by Q4, we have that t is not inhibited. Thus we can fire t in M' using the same binding such that $M' \xrightarrow{t} M_1'$, and the effect of t will be the same in both nets, which implies that m is also the difference between M_1 and M_2 in $\{p_0\} \cup t_0^\bullet$, i.e. $M_1(p_0) \setminus m = M_1'(p_0)$ and $M_1(p) \uplus m = M_1'(p)$ for all $p \in t_0^\bullet$, and therefore $M_1 \equiv_Q M_1'$.

3. : Let $M \equiv_Q M'$ and let $M' \xrightarrow{t} M_1'$. By definition of $\equiv_Q$ there exists a multiset $m \subseteq M_0(p_0)$ such that $n \cdot |\boxminus(p_0, t_0)| = |m|$ for some $n \in \mathbb{N}_0$ and $M(p_0) \setminus m = M'(p_0)$ and $M(p) \uplus m = M'(p)$ for all $p \in t_0^\bullet$. This means, by Q2, Q4, Q5, and Q9, that we can fire t_0 n times in M, i.e. if $M^{(i)}$ is M after i firings of t_0 then $M = M^{(0)}$ and $M^{(0)} \xrightarrow{t_0} M^{(1)} \xrightarrow{t_0} \cdots \xrightarrow{t_0} M^{(n)}$. Moreover, there exists multisets $m_0, m_1, \ldots, m_n$ such for all $i, 0 \leq i \leq n$ we have that $m_i \subseteq M_0(p_0)$ and $(n - i) \cdot |W| = |m_i|$ and $M^{(i)}(p) = M'(p)$ for all $p \in P \setminus (\{p_0\} \cup t_0^\bullet)$ and $M^{(i)}(p_0) \setminus m_i = M'(p_0)$ and $M^{(n)}(p) \uplus m_i = M'(p)$ for all $p \in t_0^\bullet$. In other words, $M^{(i)} \equiv_Q M'$ for all $0 \leq i \leq n$ since the size of m_i is divisible by $|W|$. We have that $m_n = \emptyset$ so $M^{(n)} = M'$, and since $M' \xrightarrow{t} M_1'$, trivially, $M^{(n)} \xrightarrow{t} M_1$ too and $M_1 \equiv_Q M_1'$.

The properties 1., 2., and 3., implies that $\equiv_Q$ is a stuttering bisimulation. So by Theorem 4, Rule Q must be correct for CTL*\X. □

6.8 CPN results

As described in Section 4.2 we have implemented the colored reduction rules C, D, E, F, I, Q, and U in `verifypn` [17, 4], which now applies these structural reductions to the CPN before it is unfolded. Our experiments include using each rule individually to compare rules directly, and two rule sequences: IUDCEFQ applying all rules, and IUC applying our best performing rules. There are many permutations of sequences in which the rules can be applied, and we chose the IUDCEFQ sequence as the permutation where we apply all our rules. We chose this as Rule I and U are the rules that reduce the CPNs the most, and having

Rule I and U early on in the rule application sequence should result in an overall faster reduction of the CPN. Since TAPAAL did not have any color reduction prior to this thesis, the *base* experiment does not perform any colored structural reductions. Recall that all experiments use the structural reductions available for P/T nets in `verifypn` as they are defined in [5] by Bønneland et al. and in [10] by van Diepen et al. This way we investigate if our CPN reduction rules contribute to an already well-developed suite of structural reductions for P/T nets. Also recall that we are using the CPNs of MCC2021 for the colored structural reduction experiments, and there are 22.080 queries in total, which is 3.680 per category.

In Table 6 we see the number of applications of each rule in each experiment, and in the Table 5 we see the percentages of queries where the given rule was applied at least once. We see that Rule C, D, E are rarely applied when used individually. This is because they remove parallel structures and dead transitions of which there are few in the MCC2021 benchmark. Rule F is used in more cases, since stable places are common. In absolute terms, Rule I is not applied many times, but percentage-wise it is applied in one third of reachability queries. Rule I is however only rarely appliedon CTL and LTL, only when these can be simplified to a reachability query. This low number of absolute number of applications, is because it can at most be applied once per query when used alone, as it removes all irrelevant parts of the net in once application. Rule U is the most-often applied rule according to the percentages. In Section 6.5 and 6.6 we showed that Rule I and U are only applicable for reachability properties. However, Rule I and U are also applied on the LTL and CTL queries whenever they can be simplified to a reachability query. Also recall that Rule U falls back to a standard pre agglomeration on non-reachability and on deadlock properties, which results in more applications on LTL queries. When the rules are used together the applicability of Rule C doubles in the reachability category, from about 4%, to about 9%, and its absolute number of applications more than doubles. Conversely, the applicability of F and Q are about halved in the IUC and IUDCEFQ experiments as compared to when used alone. However, Rule F and Q are used last in this rule sequence, which clearly reduces the number of possible applications, as there have been other reductions before these. The number of applications of Rule D, E, and I are less affected in the experiments containing other rules. We find that every rule is able to remove parts of the net which the others cannot, as all rules still has applications in the experiment which tries to apply all rules.

Table 7 shows the number of answers found in each experiment within the time limit of 4 minutes. The number of answers is shown as differences compared to the base experiment, which shows the absolute number of answers. See Appendix C where we split this table into several tables and show answers gained per model. Considering that noise may be at fault for small differences in the results, most of our rules do not seem to affect the number of answer found when used individually. Rule C is an exception, especially on ReachabilityFireability queries where it gains a total of 65 more answers. Rule U also has a positive

Experiment	C	D	E	F	I	Q	U	IUC			IUDCEFQ						
Category ╲ Rule	C	D	E	F	I	Q	U	I	U	C	I	U	D	C	E	F	Q
ReachabilityCardinality	4.3%	7.4%	5.7%	33.0%	34.7%	8.8%	43.9%	35.7%	44.4%	9.3%	35.7%	44.5%	3.5%	9.3%	9.6%	13.6%	5.2%
ReachabilityFireability	4.3%	6.8%	5.6%	40.9%	40.3%	8.1%	43.1%	40.6%	43.8%	9.0%	40.7%	44.0%	5.0%	9.0%	9.6%	15.1%	4.8%
LTLCardinality	4.3%	1.5%	5.7%	34.5%	0.0%	2.4%	6.3%	0.0%	6.3%	5.5%	0.0%	6.3%	1.2%	5.5%	6.8%	34.7%	1.5%
LTLFireability	4.3%	7.0%	5.6%	40.9%	0.0%	2.1%	6.2%	0.0%	6.2%	5.6%	0.0%	6.2%	7.0%	5.6%	6.7%	41.1%	1.3%
CTLCardinality	2.6%	0.5%	5.7%	15.2%	0.8%	1.6%	1.1%	0.8%	1.1%	2.8%	0.8%	1.1%	0.5%	2.8%	6.6%	15.1%	1.5%
CTLFireability	4.3%	1.5%	5.6%	40.9%	1.1%	1.2%	1.0%	1.2%	1.0%	4.6%	1.2%	1.0%	1.5%	4.6%	6.1%	40.2%	1.1%

Table 5: Percentages of queries where the given rule was applied at least once for the given experiment and category

Experiment	C	D	E	F	I	Q	U	IUC			IUDCEFQ						
Category ╲ Rule	C	D	E	F	I	Q	U	I	U	C	I	U	D	C	E	F	Q
ReachabilityCardinality	159	272	656	4820	1276	323	18962	1347	18202	914	1364	18715	128	1058	728	1297	191
ReachabilityFireability	160	249	592	5792	1484	298	16688	1505	15687	891	1525	16137	185	1026	793	1453	178
LTLCardinality	160	54	656	4960	0	88	2795	0	2861	311	0	3232	43	423	698	5567	56
LTLFireability	160	258	695	5792	0	79	2963	0	3043	333	0	3485	259	483	735	6397	49
CTLCardinality	95	20	656	2167	30	59	493	31	534	133	31	538	18	133	692	2201	56
CTLFireability	160	54	164	5792	42	44	464	43	503	201	43	530	56	211	184	5772	39

Table 6: Number of applications of the given rule in the given experiment and category

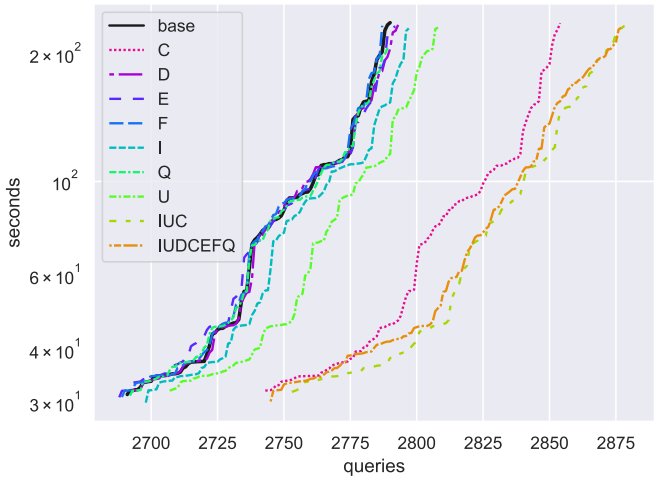
Category ╲ Experiment	base	C	D	E	F	I	Q	U	IUC	IUDCEFQ
ReachabilityCardinality	3113	+20	-3	-1	-1	+2	-4	+2	+29	+26
ReachabilityFireability	2796	+65	+3	+1	-1	+8	+2	+19	+87	+84
LTLCardinality	3205	+9	+0	+0	+0	+0	+0	-2	+9	+0
LTLFireability	3068	+10	-1	+5	+0	+1	+0	+3	+15	+9
CTlCardinality	2903	-3	+0	-3	-2	-3	-3	+0	+0	-2
CTLFireability	2322	+21	+1	+0	+1	+1	+1	+1	+22	+21
Sum	17407	+122	0	+2	-3	+9	-4	+23	+162	+138

Table 7: The number of answer found by the base experiment and the number of answers found compared to the base experiment

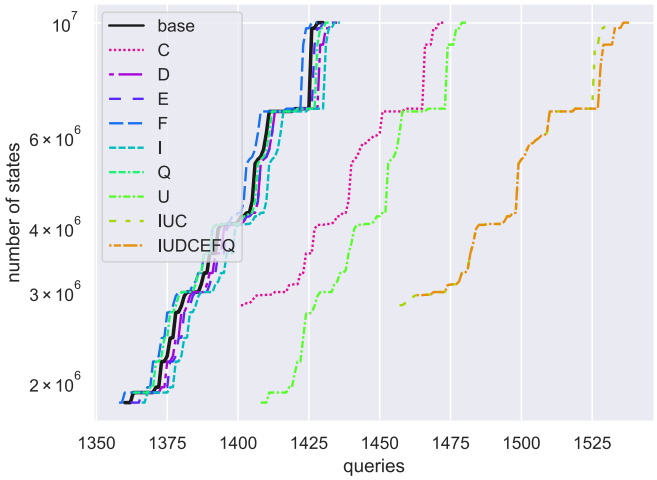
effect on the number of ReachabilityFireability answers found. Surprisingly, the experiment IUDCEFQ finds about the same number of answers as experiment C, with only some more in ReachabilityFireability but less in LTLCardinality. The experiment IUC performs better than IUDCEFQ with a few more answers across all reachability and LTL categories, and IUC is hence our best performing experiment. The new answers are primarily found on the model PolyORBNT when using Rule C and to a small extent on SharedMemory when using Rule U.

Figure 23 shows various measures for the ReachabilityFireability queries and how each experiments performs. We have chosen to highlight results from the ReachabilityFireability category, as this is the category where our rules are most applicable, and we have obtained the best results. Results for the other categories are more modest, but are still performing better than or equal to *base*. They can be found in Appendix C. We will not show the results from measuring colored reduce time, as the time spent using the colored reductions were neglible in comparison to the other metrics, such as unfold time and applying structural reductions on the P/T nets. This is due to the size of the CPNs, where the structures are generally much smaller than the P/T nets, and are as such faster to apply structural reduction rules to. For reference, for most models it took on the order of 10^{-2} seconds to apply the colored structural rules, and the graphs for this metric can also be found in Appendix C

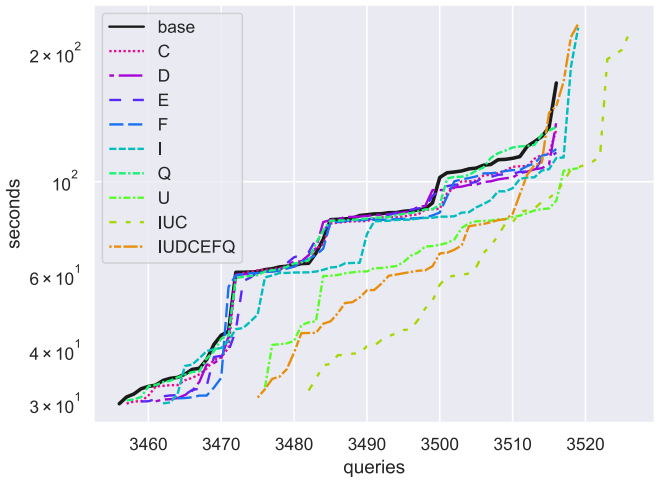
The Subfigure (a) in Figure 23 shows that experiment C, IUC, and IUDCEFQ uses less total time and thus answers more queries. Rule U and I also uses slightly less time than the baseline, but the other experiments are similar to the baseline. Regarding the size of the state space shown in Subfigure (b), Rule F seem to increase it compared to the baseline, despite the fact that Rule F only removes redundant places that never disables or inhibits transitions. Experiments U, IUC, and IUDCEFQ all significantly improve the unfold time as seen in subfigure (c). These three experiments are also the ones decreasing the time spent on structural reductions of the unfolded net according to subfigure (d). Subfigure (e) shows the time spent verifying the property, and experiments using Rule C are performing the best here. As seen on subfigure (f), Rule C barely removes any structure from the CPN, but what it does remove seem to have a big impact on the number of answers.



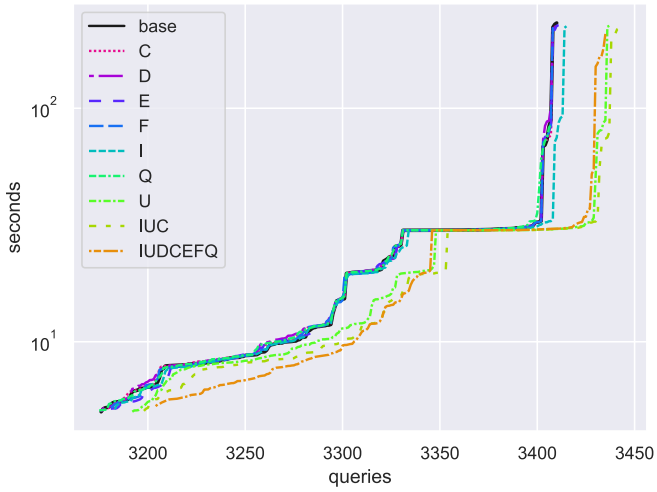
(a) Total time



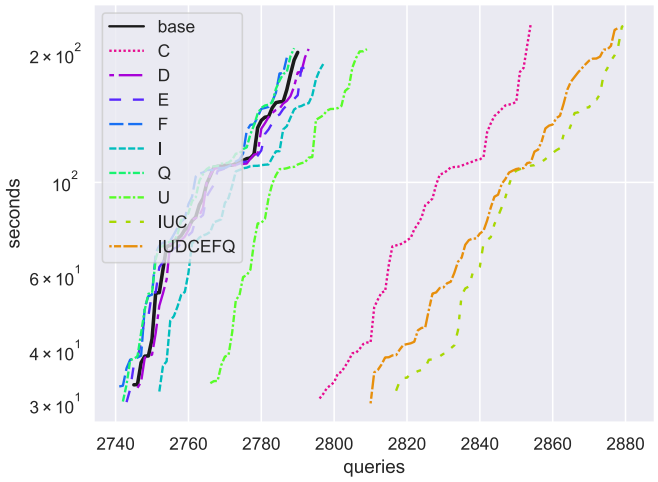
(b) State space size



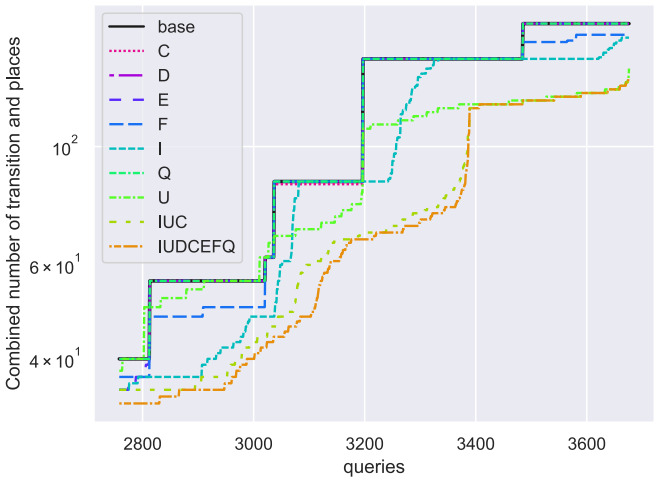
(c) Unfold time



(d) Reduce time



(e) Verification time



(f) CPN Reduced size

Figure 23: Reachability fireability test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.

As mentioned, similar graphs can be found for the other categories in Appendix C.2. But we will now summarise the results for the other categories. The results are similar for the reachability cardinality although less pronounced. One thing to note, is that Rule Q has a small negative impact on verification time in all categories. On LTL and CTL, Rule U can rarely be applied, so the biggest reductions are made by Rule I. It is, however, still experiments using Rule C which reduces the state space the most, no matter the category. Compared to the baseline, experiment E spends much less time spent on structural reductions of the unfolded net, except on CTL fireability queries, while IUDCEFQ spends less CTL cardinality and more time on LTL fireability queries. IUDCEFQ also spends much more time on unfolding of CTL cardinality queries.

The conclusion on the results from using our colored structural reduction rules, is that Rule C, I, and U are the most benefiting rules. The main goal of applying structural rules is to obtain more answers within the given time limit, and we see from Table 7 that it is these three rules that obtains the most additional answers. Unfortunately, Rule I and U are only correct for reachability queries, which limits their usefulness. The benefit from using these new colored rules are primarily time saving in unfolding- and application of rules in P/T nets.

7 Discussion

In this section we will discuss how our results have been affected by the existing P/T net reductions, reduction rules which we have not created a colored variant of, and a canonical form for CPNs which may benefit future structural reductions.

7.1 CPN reductions vs P/T net reductions

With the exception of Rule U, the colored structural reduction rules presented in this thesis have P/T net variants either from [5] by Bønneland or from [28] by Thierry-Mieg, and each of those P/T net variants is implemented in `verifypn` already. In our CPN experiments we compared *base* which uses the existing P/T net reductions in `verifypn` to IUDECFQ which uses both the existing P/T net reductions and our new CPN reductions. This comparison shows what our new CPN reductions offer to the existing `verifypn` engine. We find that CPN reductions indeed reduce the size of the CPNs and thus the unfolding time and the size of the unfolded net. The reduction rules are also able to affect the query simplification in ways the P/T net reductions cannot, since query simplification happens after unfolding of the net but before P/T net reductions. In many cases, however, our colored variants of the reduction rules simply performs a similar reduction at a higher abstraction level in an earlier phase of the engine. After the unfolding, similar P/T net reductions are performed but with finer granularity. This may be one reason why our colored Rule D, E, F, I, and Q does not help

reduce the final state space compared to our baseline. They reduce the time spent unfolding, but does not create a smaller net in the end.

Specifically for Rule F, another reason for the lack of additional answers is that the unfolder of `verifypn` does not unfold stable places, i.e. places which never gains or loses tokens according to its arcs. These stable places are also what Rule F removes from the CPN. Hence, Rule F barely affects the unfolding time, and simply only enables other colored structural reductions.

7.2 Additional reductions rules

There are P/T reductions rules from [5, 10, 28] which we have not defined a colored variant of due to time constraints. However, we believe many of them have a colored variant. We attempted to define a colored variant of Rule M by van Diepen et al. in [10]. Rule M uses fixed-point iteration to find and remove all places which will never gain or lose tokens. The transitions which the removed places disable or inhibit in the initial marking are also. Rule M is therefore a more general rule than Rule E.

However, we found that generalizing Rule M to CPNs requires checking all bindings of all transitions potentially multiple times while keeping track of all colors in all places. Therefore, colored Rule M is computationally infeasible, and we opted to define the overapproximation that is colored Rule E.

7.3 Canonical form

Some colored structural reduction rules, e.g Rule D, involves checking if two guards are semantically equivalent. This check is significantly easier if the CPNs have a canonical form for their guard, and arc, expressions. In our definition of CPNs in Section 3 we do not allow arc expressions, but instead move such constraints into the guard on the transition. Instead of expressions on the arcs, we then have multisets of tuples of variables, which is easier to compare against one another. Hence, we need only a canonical form for guard expressions, and we present such a conceptual canonical form in Appendix D, including an algorithm to rewrite any guard expression to its canonical form. Notice that we do not prove the canonical form to be canonical, but merely a concept. We did not finish the work on the canonical form, because the MCC2021 benchmark models are extremely well-formed already, and we did not find cases, where the canonical form prevented the applicability of our reductions. However, a user of TAPAAL is not guaranteed to create well-formed CPNs and a canonical form may still be beneficial.

8 Conclusion

We presented further work on structural reductions and in particular introduced several structural reductions for colored Petri nets. We have implemented these new structural reductions in a new colored structural reduction phase

of the model checking tool TAPAAL [8]. For P/T nets we presented a more general version of the structural reduction rule, Rule C, previously described by Bønneland et al. in [5] which removes parallel places. We also presented Rule S, an atomic free agglomeration which agglomerates one consumer at a time. Rule S is the counterpart to the atomic post agglomeration, Rule R, by van Diepen et al [10]. For CPNs we presented colored variants of Rule D, E, F, and I described in [5] by Bønneland et al. and Rule Q described initially by Thierry-Mieg in [28]. We also presented colored variants of aforementioned Rule C and Rule S for P/T nets, however, the colored variant of Rule S fixes $k = 1$ due to better performance and we call it Rule U. Additionally, we proved the correctness of each presented rule for CTL* or a fragment thereof. We also gave counter-examples to show why some rules are not correct for some fragments of CTL*.

The presented rules was implemented in the model checking engine of TAPAAL to complement the existing P/T net reduction, and then experimentally tested on the P/T nets and CPNs from the Model Checking Contest [21] 2021 edition using all categories of queries. We found that Rule C for P/T nets did not provide any difference over the previous Rule C, despite being more general in theory. On the other hand, Rule S allowed us to find 393 more answers combined across all categories when added to the existing rules in TAPAAL. On CPNs, colored Rule C alone gave us an additional 122 answers across all categories, primarily on the model PolyORBNT. Rule U significantly reduced models like FamilyReunion, but only resulted in a few additional answers overall. While Rule D, E, F, and Q reduced the CPNs, they did not affect the verification overall. The best sequence of rules found was IUC with 162 answers more than the baseline in total. The experiment IUC reduced the time spent unfolded and reducing the net with non-colored structural reductions. Since Rule S and U are not applicable on LTL and CTL, our results were more modest on these categories. In conclusion, our CPN reduction rules do not exceed the existing P/T net rules, as they perform similar reductions earlier.

9 Future work

In this section we will briefly describe potential future work. As mentioned in Section 7, we believe many of the P/T net reductions from [5, 10, 28] can be generalized to CPNs, and that would be an obvious thing to do next. However, based on our observations, we believe that only some of them are suited for CPN reductions while others contribution is neglible. We therefore recommend thoroughly experimenting with each rule to test if it most beneficial as a CPN reduction, P/T net reduction, or both.

Next we would like to discuss potential future work regarding combinatorial agglomerations, which is an alternative to atomic agglomerations like Rule S, and an extension of Rule Q. The following subsections are dedicated to those two topics.

9.1 Rule S variations

While we did explore some variations of pre- and free agglomerations, there are more variations that could be worth experimenting with. Our experiments with translating the k-scaling that sets Rule S apart over to CPNs produced generally worse results than those of Rule U, but this is not necessarily universally the case. It could warrant further exploration of what situations it is worth using in.

An alternate approach to the k-scaling could be to apply it without the atomic agglomeration, instead relying on being able to identify and produce transitions for every permutation of producers and consumers that can be matched together. This would limit its application to the situations where it is likely the most impactful, but the question still remains if the state space reduction is worth the potentially large increase in transition count and reduction time.

Another agglomeration that was not touched upon is post agglomeration. Like the free agglomeration of Rule S, post agglomeration can also be made atomic such as was done with Rule R in [10], which can most likely also be translated to CPNs. It could also be tested with the types of variations as were tried with Rule S.

9.2 Rule Q extension

We presented Rule Q in Section 6.7 which finds places that can safely be emptied by firing its sole consumer, without affecting the satisfiability of the property. The presented Rule Q requires that the place being emptied, p_0, has the same type, as all of the places in the post set of the firing transition, t_0. Moreover, the expression on the arc from p_0 to t_0, has to be the exact same as on the arc from t_0 to any $p \in t_0^\bullet$. These are the conditions Q8 and Q9 in Figure 21. However, we believe that we can relax these conditions for reachability queries, by allowing a different set of tokens being produced to any p, as long as we do not disable future behaviour. A concrete example of an application of this extension is shown in Figure 24 where neither Q8 nor Q9 holds, if we fix *Weight_Left_Wheel* as p_0. We can notice, that there are only two colours that can be produced to *stp1* from firing *SampleLW*, either *Weight0* or *Weight1*. The only consumer of the place *stp1* has a guard which only valid binding of the variable w is *Weight0*. We can therefore safely empty the place *Weight_Left_Wheel* and put one token of *Weight0* into *stp1*. Then we would also be able to remove both *Weight_Left_Wheel* and *SampleLW* with rule I, and rewrite the arc from *stp1* to $CCT3$, such that the guard becomes True, and simply takes a *1'dot* from *stp1* instead. This specific example occurs in the CPN AirplaneLD-COL-0200 in MCC2021 using query 6 of the ReachabilityCardinality category after having applied all of our proposed colored reduction rules on it,. The guard has more conditions in the CPN, which has been left out in this example, but the idea holds.

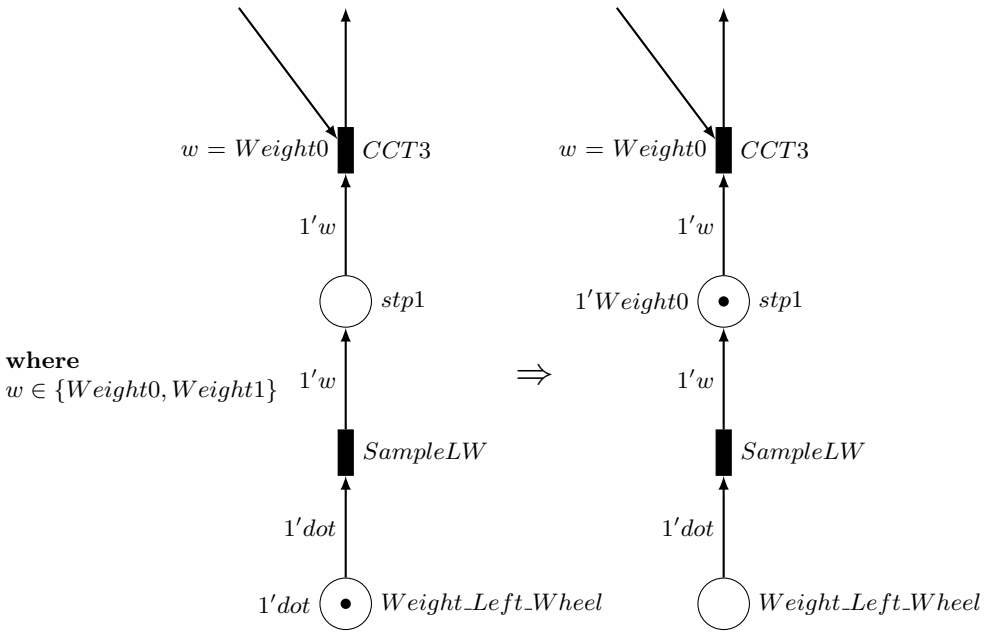


Figure 24: Rule Q: Preemptive firing extension - single valid color

Acknowledgement We would like to thank Jiří Srba and Peter G. Jensen for supervising.

References

[1] Gérard Berthelot. 'Checking properties of nets using transformation'. In: *Applications and Theory in Petri Nets*. 1985.

[2] Gérard Berthelot, Gérard Roucairol and Rüdiger Valk. 'Reductions of nets and parallel programs'. In: *Net theory and applications*. Springer, 1980, pp. 277–290.

[3] M.C. Browne, E.M. Clarke and O. Grümberg. 'Characterizing finite Kripke structures in propositional temporal logic'. In: *Theoretical Computer Science* 59.1 (1988), pp. 115–131. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(88)90098-9. URL: https://www.sciencedirect.com/science/article/pii/0304397588900989.

[4] Frederik Bønneland et al. 'Simplification of CTL Formulae for Efficient Model Checking of Petri Nets'. In: Jan. 2018, pp. 143–163. ISBN: 978-3-319-91267-7. DOI: 10.1007/978-3-319-91268-4_8.

[5] Frederik M. Bønneland et al. 'Stubborn versus structural reductions for Petri nets'. In: *Journal of Logical and Algebraic Methods in Programming* 102 (2019), pp. 46–63. ISSN: 2352-2208. DOI: https://doi.org/10.

1016/j.jlamp.2018.09.002. URL: https://www.sciencedirect.com/science/article/pii/S235222081830035X.

[6] Edmund M. Clarke and E. Allen Emerson. 'Design and synthesis of synchronization skeletons using branching time temporal logic'. In: *Logics of Programs*. Ed. by Dexter Kozen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 52–71. ISBN: 978-3-540-39047-3.

[7] Edmund M Clarke et al. *Handbook of model checking*. Vol. 10. Springer, 2018.

[8] A. David et al. 'TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets'. In: *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*. Vol. 7214. LNCS. Springer-Verlag, 2012, 492–497.

[9] Martin Didriksen et al. 'Automatic Synthesis of Transiently Correct Network Updates via Petri Games'. In: *Application and Theory of Petri Nets and Concurrency*. Ed. by Didier Buchs and Josep Carmona. Cham: Springer International Publishing, 2021, pp. 118–137. ISBN: 978-3-030-76983-3.

[10] Nicolaj Ø. Jensen Jesper A. van Diepen and Mathias M. Sørensen. 'Improvements to Structural Reductions of Petri Nets with Inhibitor Arcs'. In: (2021).

[11] E. Allen Emerson and Joseph Y. Halpern. '"Sometimes" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic'. In: *J. ACM* 33.1 (1986), 151–178. ISSN: 0004-5411. DOI: 10.1145/4904.4999. URL: https://doi.org/10.1145/4904.4999.

[12] S. Evangelista, S. Haddad and J. F. Pradat-Peyre. 'Syntactical Colored Petri Nets Reductions'. In: *Automated Technology for Verification and Analysis*. Ed. by Doron A. Peled and Yih-Kuen Tsay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 202–216. ISBN: 978-3-540-31969-6.

[13] Serge Haddad and Jean-François Pradat-Peyre. 'Efficient reductions for LTL formulae verification'. In: *Paris, France* (2004).

[14] Serge Haddad and Jean-François Pradat-Peyre. 'New Efficient Petri Nets Reductions for Parallel Programs Verification'. In: *Parallel Process. Lett.* 16 (2006), pp. 101–116.

[15] Lasse Jacobsen et al. 'Verification of Timed-Arc Petri Nets'. In: *SOFSEM 2011: Theory and Practice of Computer Science*. Ed. by Ivana Černá et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 46–72. ISBN: 978-3-642-18381-2.

[16] Jonas Jensen, Thomas Nielsen and Lars Oestergaard. *Petri Nets with Discrete Variables*. Jan. 2011.

[17] Jonas F. Jensen et al. 'TAPAAL and Reachability Analysis of P/T Nets'. In: *Transactions on Petri Nets and Other Models of Concurrency XI*. Ed. by Maciej Koutny, Jörg Desel and Jetty Kleijn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 307–318. ISBN: 978-3-662-53401-4. DOI: 10.1007/978-3-662-53401-4_16. URL: https://doi.org/10.1007/978-3-662-53401-4_16.

[18] Jonas Finnemann Jensen et al. 'TAPAAL and Reachability Analysis of P/T Nets'. English. In: *Transactions on Petri Nets and Other Models of Concurrency XI*. Lecture Notes in Computer Science. Germany: Springer, 2016, pp. 307–318. ISBN: 978-3-662-53400-7. DOI: 10.1007/978-3-662-53401-4_16.

[19] Kurt Jensen. 'Coloured Petri nets'. In: *Petri Nets: Central Models and Their Properties: Advances in Petri Nets 1986, Part I Proceedings of an Advanced Course Bad Honnef, 8–19 September 1986*. Ed. by W. Brauer, W. Reisig and G. Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 248–299. ISBN: 978-3-540-47919-2. DOI: 10.1007/BFb0046842. URL: https://doi.org/10.1007/BFb0046842.

[20] Kurt Jensen and Grzegorz Rozenberg. *High-level Petri nets: theory and application*. Springer Science & Business Media, 2012.

[21] F. Kordon et al. *Complete Results for the 2020 Edition of the Model Checking Contest*. http://mcc.lip6.fr/2021/results.php. 2021. (Visited on 2021).

[22] Alberto Martelli and Ugo Montanari. *Unification in linear time and space: A structured presentation*. Istituto di Elaborazione della Informazione, Consiglio Nazionale delle Ricerche, 1976.

[23] T. Murata. 'Petri nets: Properties, analysis and applications'. In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580. DOI: 10.1109/5.24143.

[24] Emmanuel Paviot-Adet et al. *LTL under reductions with weaker conditions than stutter-invariance*. 2021. arXiv: 2111.03342 [cs.CL].

[25] Carl Adam Petri. 'Communication with automata'. In: (1966).

[26] Amir Pnueli. 'The temporal logic of programs'. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32.

[27] Denis Poitrenaud and Jean-François Pradat-Peyre. 'Pre- and Post-agglomerations for LTL Model Checking'. In: vol. 1825. June 2000, pp. 387–408. ISBN: 978-3-540-67693-5. DOI: 10.1007/3-540-44988-4_22.

[28] Yann Thierry-Mieg. 'Structural Reductions Revisited'. In: *Application and Theory of Petri Nets and Concurrency*. Ed. by Ryszard Janicki, Natalia Sidorova and Thomas Chatain. Cham: Springer International Publishing, 2020, pp. 303–323. ISBN: 978-3-030-51831-8.

A Rule T: Pre agglomearation

Rule T in Figure 25 is a pre agglomeration [28].

Theorem 15. *[27] A pre agglomeration (Rule T) described in Figure 25 is correct for LTL\X.*

B Rule S: Colored atomic k-scaled free agglomeration

This appendix covers the expansion of the colored Rule U to include the same k-scaling that was applied to uncolored nets in Rule S. This variant was not in-

Precondition	Update
Fix place p_0 s.t.: T1) $(\{p_0\} \cap places(\varphi) = \emptyset$ T2) $(p_0^\bullet \cup {}^\bullet p_0) \cap transitions(\varphi) = \emptyset$ for all $h \in {}^\bullet p_0$ and $f \in p_0^\bullet$: T3) $M_0(p_0) < \boxminus(p_0, f)$ T4) ${}^\bullet p_0 \cap p_0^\bullet = \emptyset$ T5) $({}^\bullet h)^\bullet = \{h\}$ T6) $h^\bullet = \{p_0\}$ T7) ${}^\bullet h \cap places(\varphi) = \emptyset$ T8) $p_0^\circ = {}^\circ h = ({}^\bullet h)^\circ = \emptyset$ T9) $\boxplus(h, p_0) = \boxminus(p_0, f)$	Create transition $\langle hf \rangle$ for all $h \in {}^\bullet p_0$ and $f \in p_0^\bullet$ s.t. for all $p \in P \setminus \{p_0\}$: UT1) $\boxminus(p, \langle hf \rangle) = \boxminus(p, h) + \boxminus(p, f)$ UT2) $\boxplus(\langle hf \rangle, p) = \boxplus(f, p)$ UT3) $I(p, \langle hf \rangle) = I(p, f)$ and UT4) Remove ${}^\bullet p_0$, $p_0^\bullet$ and p_0

Figure 25: Rule T: Pre agglomeration.

cluded in the final experiments as it showed generally worse results than variants without k-scaling.

Theorem 16. *Rule S in Figure 19 is correct for reachability without deadlock.*

B.1 Preconditions pointwise

This section will got through the preconditions of Rules T, U, and S in more detail.

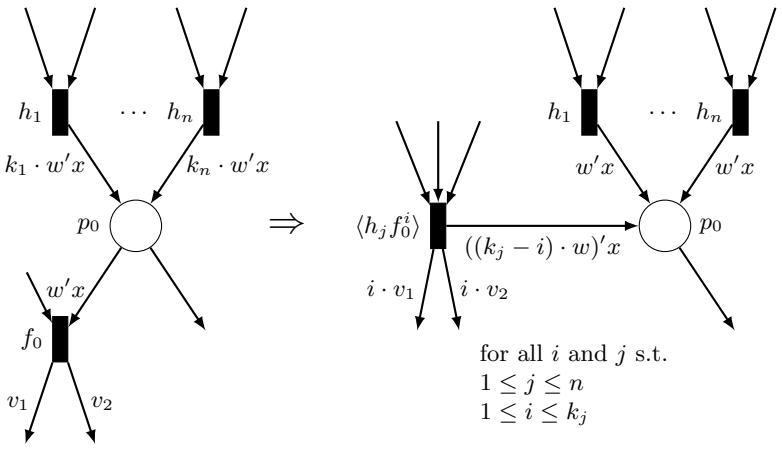
T1/U1/S1, T2/U2/S2, and T7/U8/S8 collectively establish that the places in which the delayed transitions produce and consume tokens are not in the query, and that the transitions these places may enable are not in the query.

T3/U3/S3 establishes that no firings of f_0 are possible without the previous firing of some h, as $\langle hf \rangle$ would not replicate such a firing.

T4/U4/S4 establishes that no transition both produces and consumes from p_0, as such a transition could facilitate an infinite applications of this rule producing an infinite number of additional transitions.

U5/S5 is that f_0 is a consumer of p_0. In T this is given by the "for all $f \in p_0^\bullet$" clause instead.

T5 establishes that only a single transition may consume tokens from each place in ${}^{\bullet\bullet}p_0$. This is necessary to preserve deadlock and $LTL\backslash X$ but not reachability, and as such there is no equivalent precondition in S.



Precondition	Update
Fix place p_0 and transition f_0 s.t.: S1) $(\{p_0\} \cap places(\varphi) = \emptyset$ S2) $({}^\bullet p_0 \cup p_0^\bullet \cup ({}^{\bullet\bullet} p_0)^\bullet) \cap transitions(\varphi) = \emptyset$ S3) $M_0(p_0) = \emptyset$ S4) ${}^\bullet p_0 \cap p_0^\bullet = \emptyset$ S5) $f_0 \in p_0^\bullet$ S6) $\lvert \mathbf{Supp}(\boxminus(p_0, f_0)) \rvert = 1$ and for all $h \in {}^\bullet p$: S7) $h^\bullet = \{p_0\}$ S8) ${}^\bullet h \cap places(\varphi) = \emptyset$ S9) $p_0^\circ = {}^\circ h = ({}^\bullet h)^\circ = \emptyset$ S10) $\lvert \boxplus(h, p_0) \rvert = k * \lvert \boxminus(p_0, f_0) \rvert$ S11) $k > 1 \implies {}^\bullet f_0 = \{p_0\}$ S12) $k > 1 \implies (f_0^\bullet)^\circ = \emptyset$ And for each variable $v \in ((\boxplus(h, p_0) \cup \boxminus(p_0, f_0)) \cap (\mathbf{Vars}(G(h)) \cup \mathbf{Vars}(G(f_0)))$ there exists a $p \in P \backslash \{p_0\}$ such that: S13) $v \in (\mathbf{Vars}(\boxplus(h, p)) \cup \mathbf{Vars}(\boxminus(p, f_0)))$	For all $h \in {}^\bullet p_0$, create a transition $\langle hf \rangle$ s.t. for all $p \in P \setminus \{p_0\}$, for all $i \in [1, k]$ for the k such that $\lvert \boxplus(h, p_0) \rvert = k * \lvert \boxminus(p_0, f_0) \rvert$: US1) For all $v \in \mathbf{Vars}(f_0)$, $rename(f_0, v, v')$ with some $v' \in \mathbf{Var}_{\mathcal{X}(p)} \backslash \mathbf{Vars}(h)$ then US2) $\boxminus(p, \langle h f_0^i \rangle) := \boxminus(p, h) \uplus \boxminus(p, f_0)$ US3) $\boxplus(\langle h f_0^i \rangle, p) := i * \boxplus(f_0, p)$ $\boxplus(\langle h f_0^i \rangle, p_0) := (k - i) * \boxminus(p_0, f_0)$ US4) $G(\langle h f_0^i \rangle) := G(h) \wedge G(f_0)$ US5) $I(\langle h f_0^i \rangle) := I(f_0)$ then US6) Given that $\boxplus(h, p_0) = \{\langle x_1, x_2, \ldots, x_n \rangle\}$ and $\boxminus(p_0, f_0) = \{\langle y_1, y_2, \ldots, y_n \rangle\}$ For $j \in [1, n]$ Let l be the smallest number s.t. $x_l = x_i$ holds: $rename(\langle h f_0^i \rangle, x_j, y_l), rename(\langle h f_0^i \rangle, y_j, y_l)$ then after all such transitions are made: US7) Remove f_0 US8) If $p_0^\bullet = \emptyset$, remove p_0 and all transitions in ${}^\bullet p_0 \setminus transitions(\varphi)$

Figure 26: Rule S: Atomic k-scaled free agglomeration

T6/U7/S7 establishes that no h may produce tokens in places aside from p_0, as those could cause the timing of firing h to matter independently of f, which

agglomeration would not preserve.

T8/U9/S9 establishes that no h may be inhibited, and none of the places any h produces or consumes tokens in may be inhibiting, as those could cause the timing of firing h to matter. Note that there are no such restrictions on f, as any $\langle hf \rangle$ in the reduced net is fired exactly when its f would have fired in the original net.

T9/U6/S6 is necessary because arcs on h with multiple variables can produce tokens with colors X,Y twice, and then fire the next transition with X,X and Y,Y. Agglomeration would remove that ability.

Collectively S6 + S10 establishes that all arcs into p_0, and the arc from p_0 to f_0 must all have a support of exactly one. This means that any binding of any h enables exactly one binding for f_0. If one firing of such an h produces a number of tokens that is an integer multiple k of the weight of f_0, that firing enables k firings of f_0 with the same single possible binding. Without S6 + S10, there may exist possible firings of f_0 using tokens from separate firings of h that would not exist after the agglomeration. Without S6 + S10, the ability to swap which token is bound to which variable between transitions would be lost in the agglomeration.

T10 requires all arcs into and out of p_0 to carry the same number of tokens, so any firing of some f can be matched with exactly one firing of some h. This is still valid when the token count shared by all the arcs is greater than 1 because of T9: Even if multiple different h are fired before any f, each subsequent firing of any f must consume tokens of only a single color, which can be attributed to a single firing of some h that produced tokens in that color.

T11/U11/S13 are equivalent preconditions. They ensure that an agglomeration is not allowed if it would result in a guard containing a variable that appears nowhere else on the transition. This is the only precondition on guards that is necessary.

The $k > 1$ cases of S rely on the understanding that in the logic EF/AG without deadlock, for any firing of some h, you can make a 'guess' of how many of the produced tokens are to be used by a given f, and just the possibility of guessing the correct amount is enough to preserve reachability. Rule S produces one transition for each possible 'guess'.

T places conditions on all $f \in p^\bullet$. In U and S however, there is no need for any conditions on other transitions than f_0 that might consume the tokens in any of the affected places, because to preserve correctness for EF/AG without deadlock, it is enough to preserve the possibility for any given state to be reached

without regard for preserving ways to not reach it.

As these k firings of f will all happen simultaneously, additional preconditions S11 and S12 are necessary to ensure agglomeration is not done in cases where the relative timing of the k firings of f matters. Preconditions S11 and S12 are only applicable to $k > 1$ cases, as they are trivially true for $k = 1$ because f firing number 1 and f firing number k are the same firing, so there can't be other transition firings between them.

S11 disallows f_0 from having input arcs from other places than p_0 in the $k > 1$ case. Such arcs can make possible situations where f_0 is able to fire (k-i) times without being able to fire k times, which the agglomeration would remove.

S12 disallows f_0 from placing tokens in inhibiting places in the $k > 1$ case. This covers cases where f_0 is inhibited by its own postset, and for the case of weighted inhibitors, where k firings of f_0 at once can activate weighted inhibitors that k individual firings interleaved with other transition firings would not necessarily do.

UT1/UU1/US1 prevents introduction of name clashes by renaming all the variables on one of the transitions to variables not on the other. The consumer is chosen for this, as it is later removed by UT6 anyway.

Updates 2 to 5 for both T and U give $\langle hf \rangle$ all the arcs of the transitions they replace, with the arcs to and from p_0 treated as canceling each other out.

The same holds for US2, US4 and US5, but US3 and US3-2 instead give $\langle hf \rangle$ arcs that combine to the same effect as firing h once and f_0 i times.

UT6/UU6/US6 renames different variables that refer to the same color within the same token into being the same variable. An alternate way this could be represented is by adding guards $x_i = y_i$ for all $i \in [1, n]$. See Figure 20

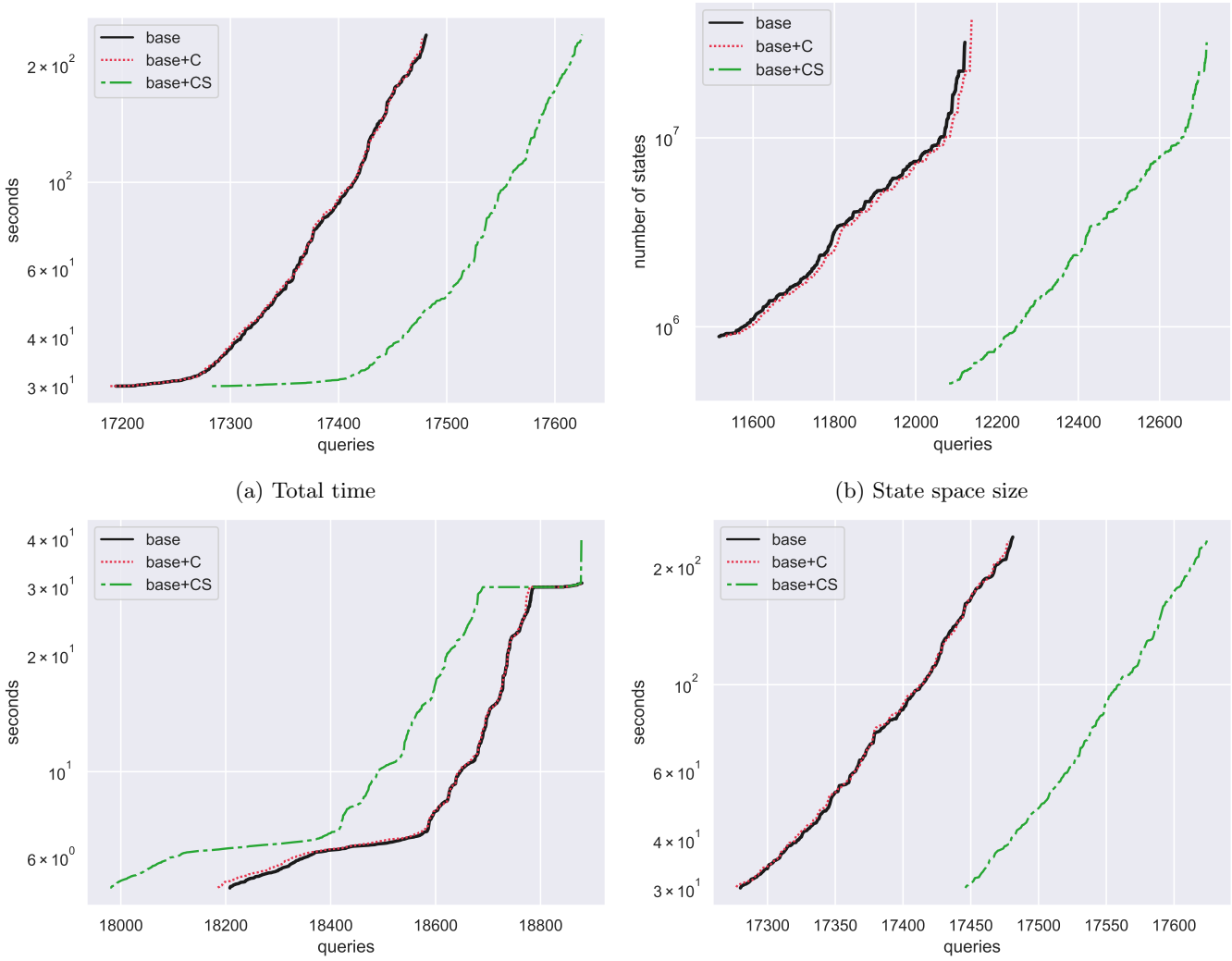
UT7/UU7/US7, and US8, remove all places and transitions that are made redundant by the new transitions.

C Additional figures

In this appendix, we show additional figures and tables generated from our testing, that we did not fit into the main text. In Appendix C.1 we show the graphs on the metrics measured during testing of our P/T reduction rules of the other categories. The next Appendix C.2 also shows additional graphs of the metrics on the other categories, but for our colored rules testing. Then in Appendix C.3 we show which models we have obtained additional answers in, for our colored rules testing. Similarily in Appendix C.4 we show additional answers by model for our P/T reduction rule testing.

C.1 P/T net reduction graphs

This section contains additional figures of the performance of our P/T net reduction rules for the various categories. The total time spent, time spent on reduction, time spent on verification, and state space size can be seen on Figure 28 (ReachabilityCardianlity), Figure 30 (LTLCardianlity), Figure 32 (LTLFireability), Figure 32 (CTLCardinality), and Figure 32 (CTLFireability). The figure with ReachabilityFireability results is found in the main text, Section 5.3.



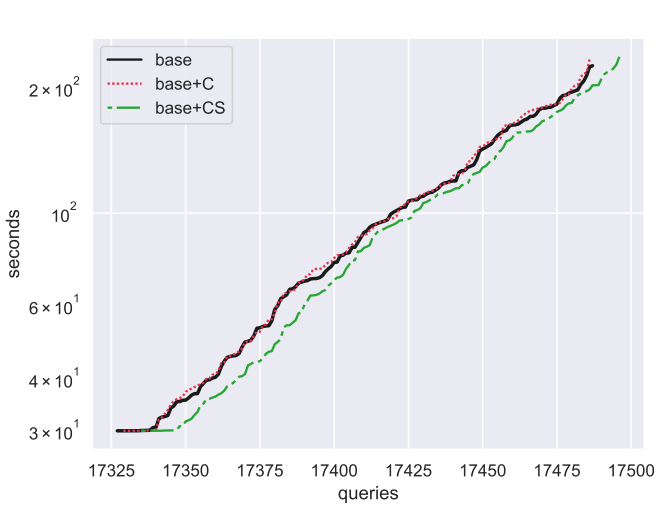
(a) Total time

(b) State space size

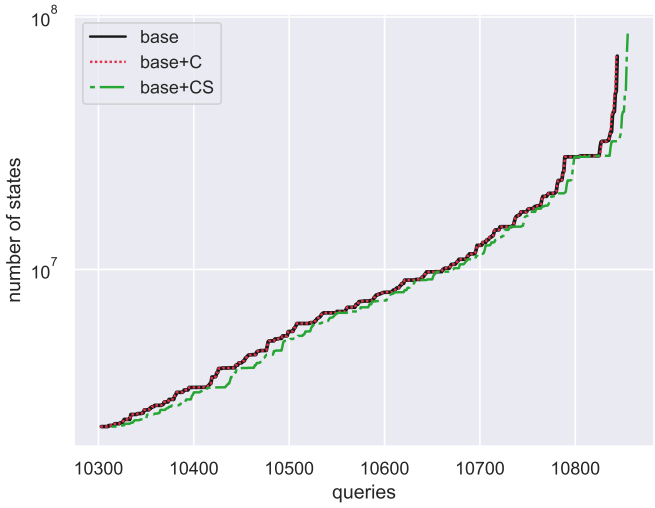
(c) Reduce time

(d) Verification time

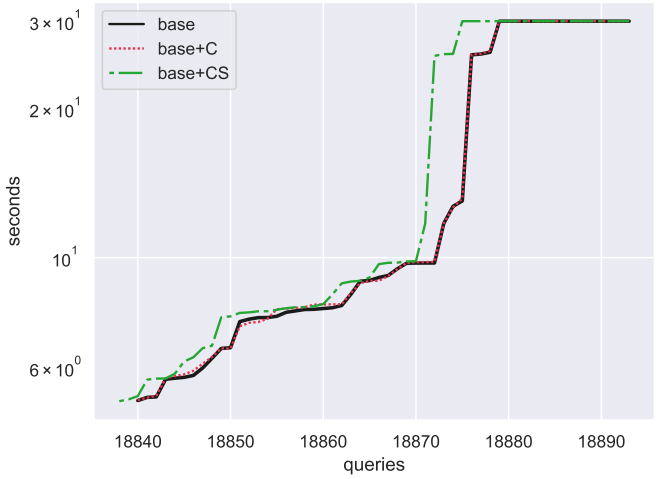
Figure 28: Reachability cardinality test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.



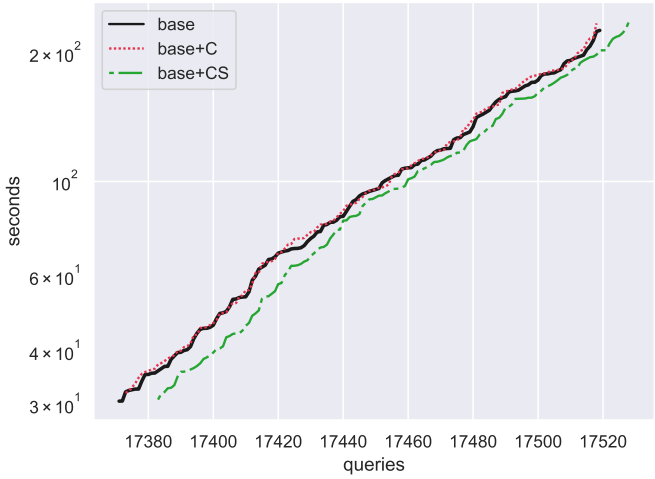
(a) Total time



(b) State space size

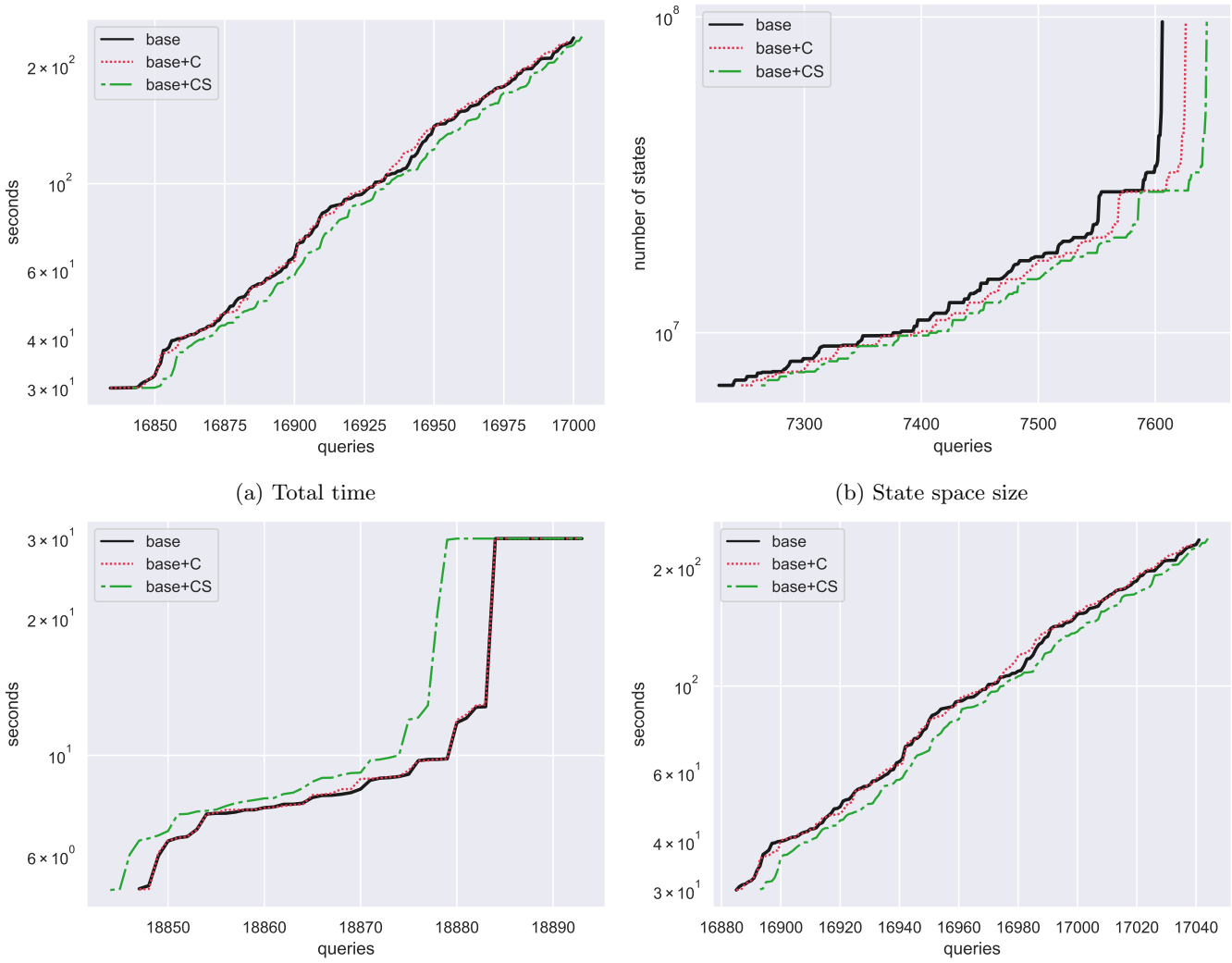


(c) Reduce time



(d) Verification time

Figure 30: LTL cardinality test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.



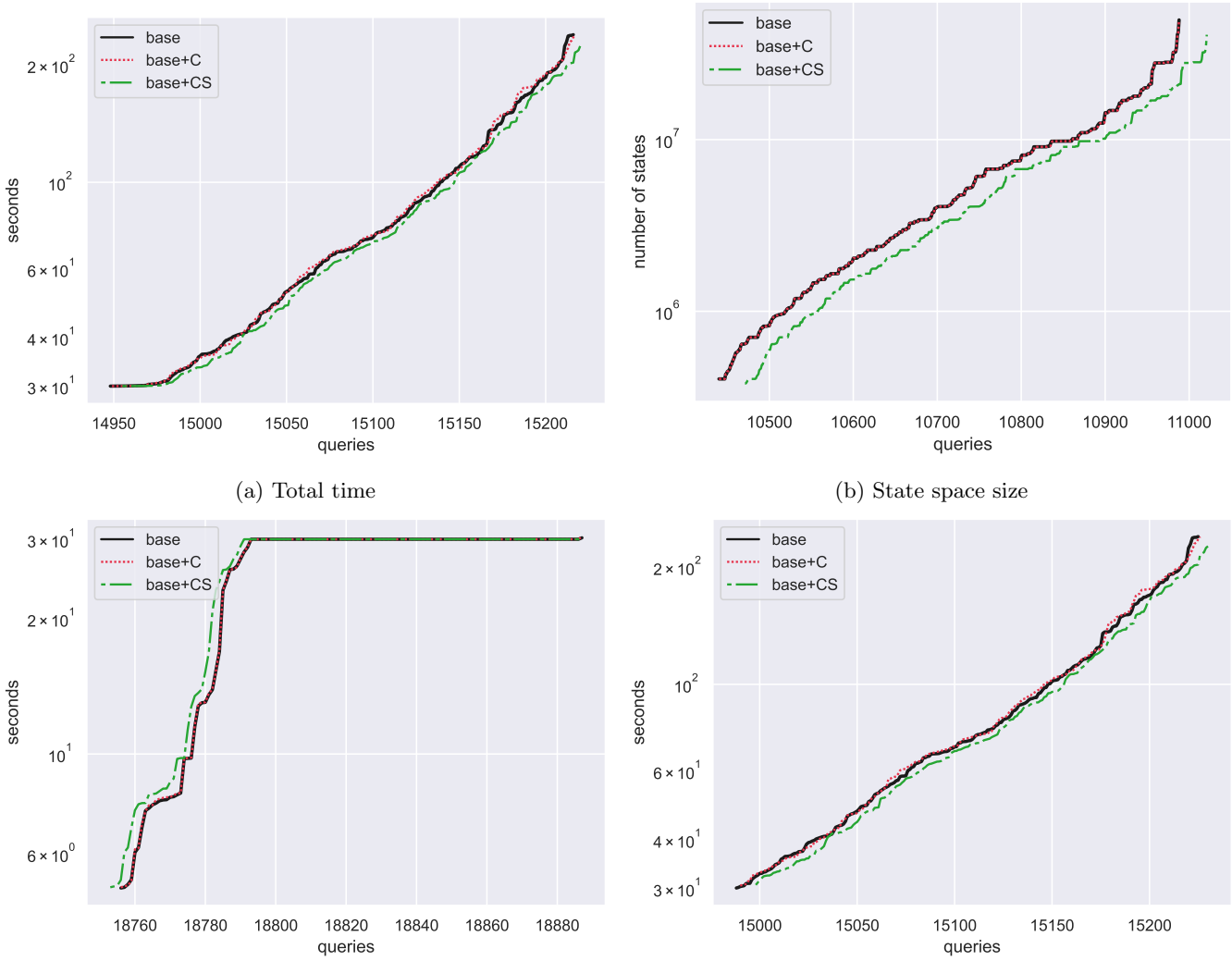
(a) Total time

(b) State space size

(c) Reduce time

(d) Verification time

Figure 32: LTL fireability test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.



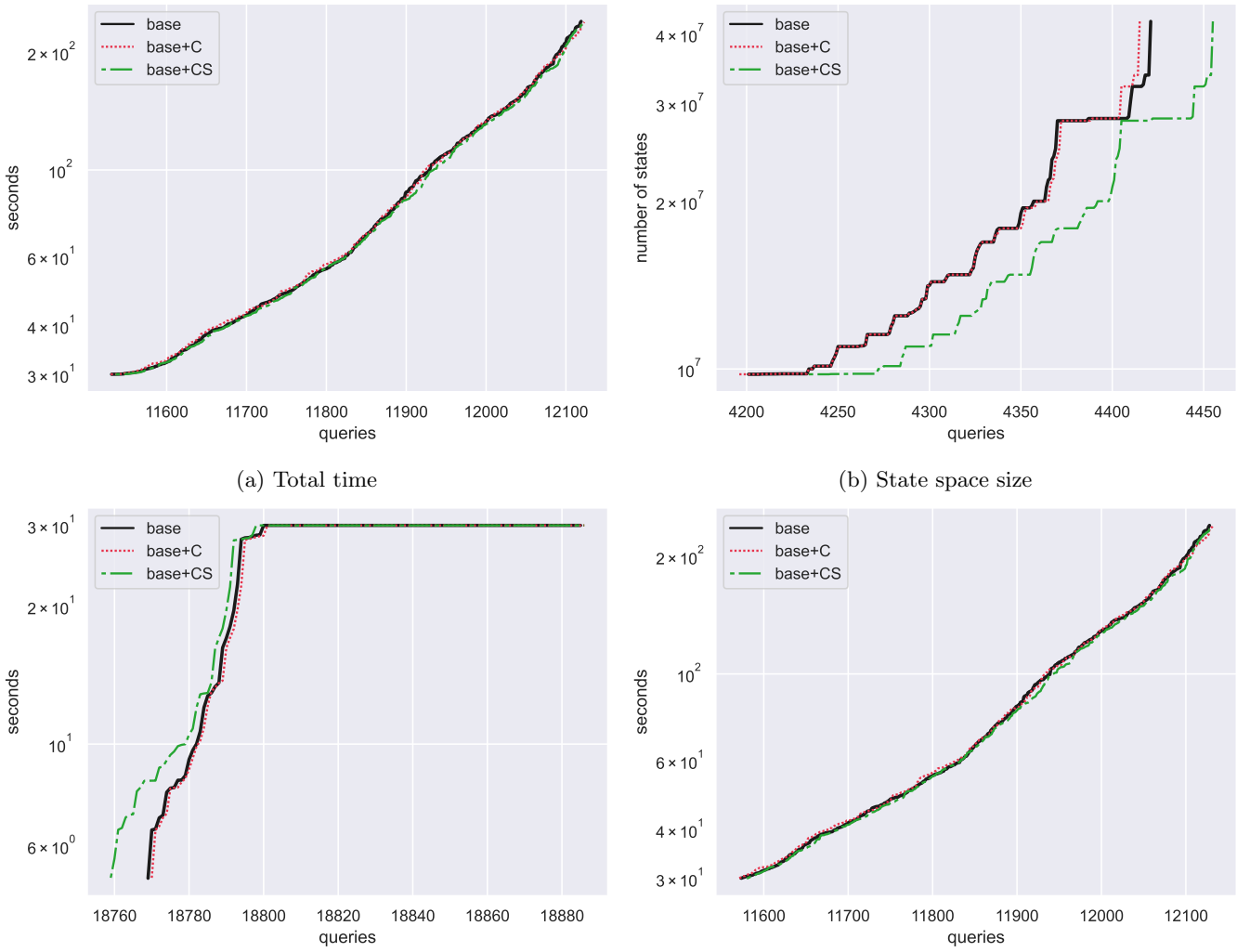
(a) Total time

(b) State space size

(c) Reduce time

(d) Verification time

Figure 34: CTL cardinality test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.



(a) Total time

(b) State space size

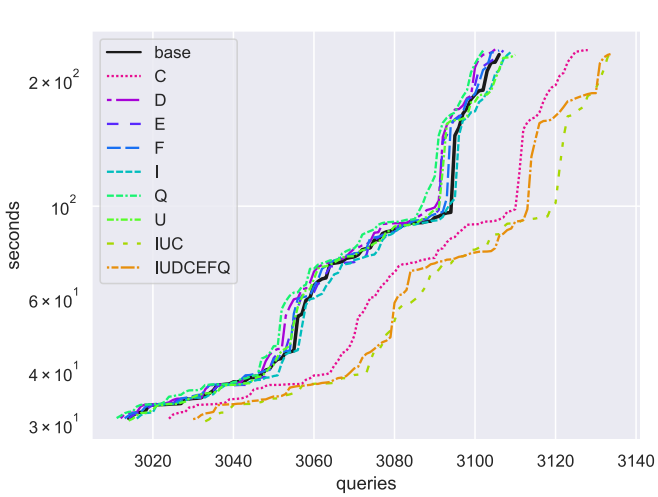
(c) Reduce time

(d) Verification time

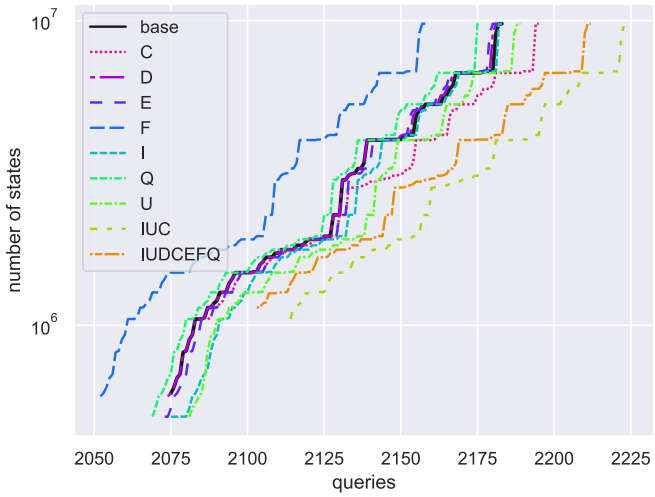
Figure 36: CTL fireability test results for P/T nets. (a) Total time spent in the reduction- and verification phase for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in reduction phase for each experiment, > 5 sec. (d) Time spent in verification phase for each experiment, > 30 sec. All data points are sorted by size.

C.2 CPN reduction graphs

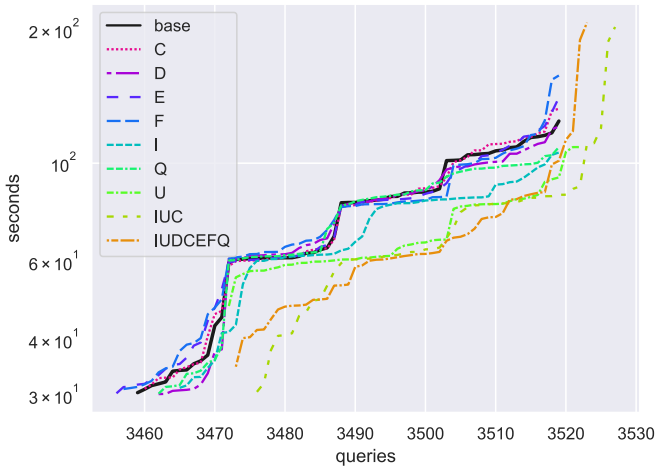
This section contains additional figures of the performance of our CPN reduction rules for the various categories. Various performance measures can be seen on Figure 38 (ReachabilityCardianlity), Figure 40 (LTLCardianlity), Figure 42 (LTLFireability), Figure 42 (CTLCardinality), and Figure 42 (CTLFireability). The figure with ReachabilityFireability results is found in the main text, Section 6.8. Figure 48 shows the time spent applying color reduction rules in the CPNs, for all categories.



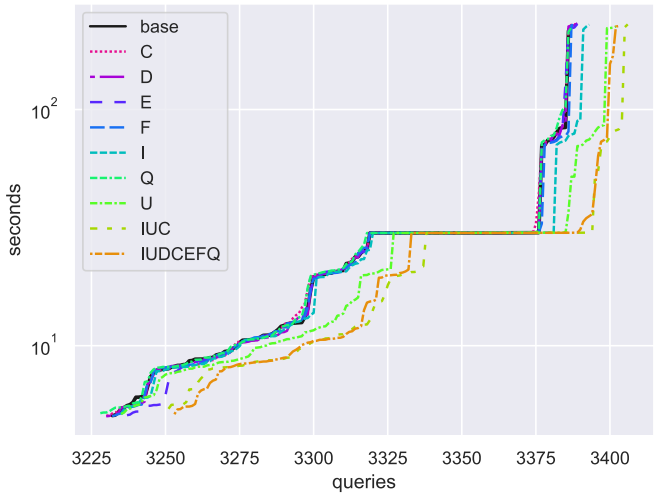
(a) Total time



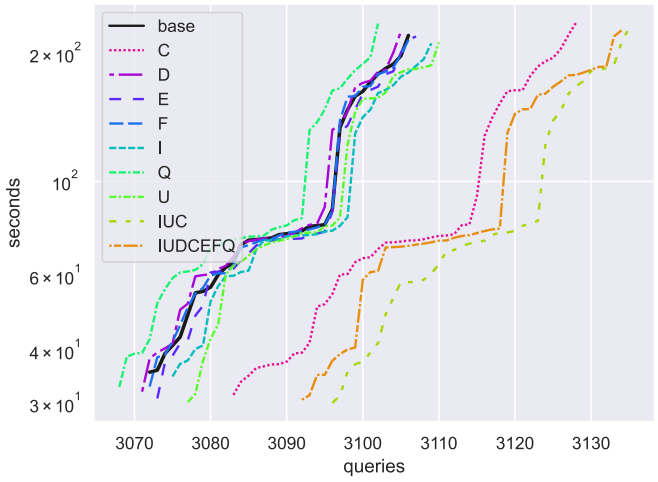
(b) State space size



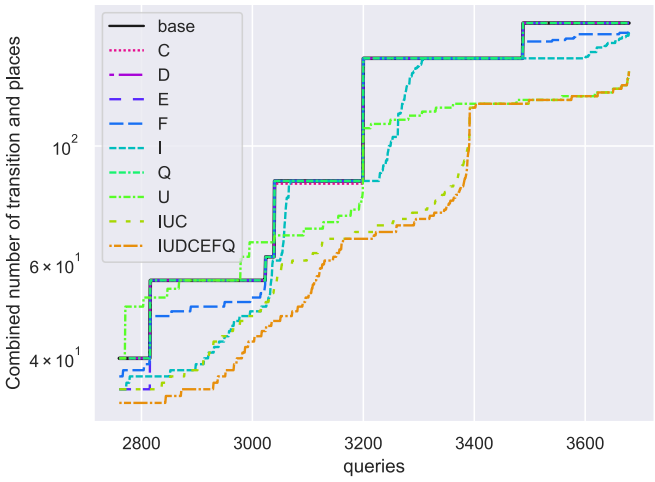
(c) Unfold time



(d) Reduce time

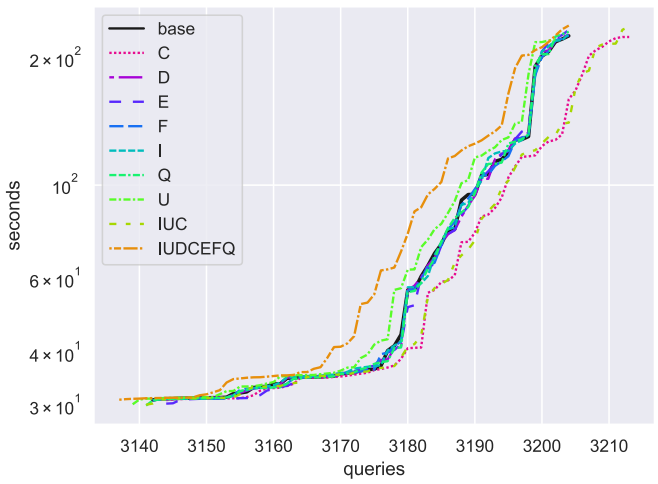


(e) Verification time

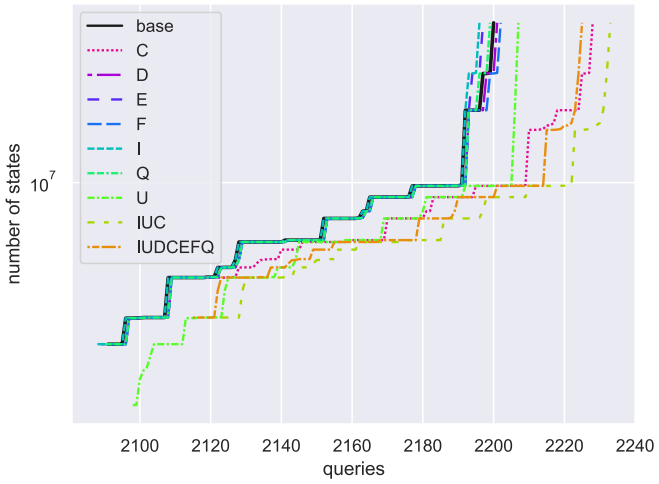


(f) CPN Reduced size

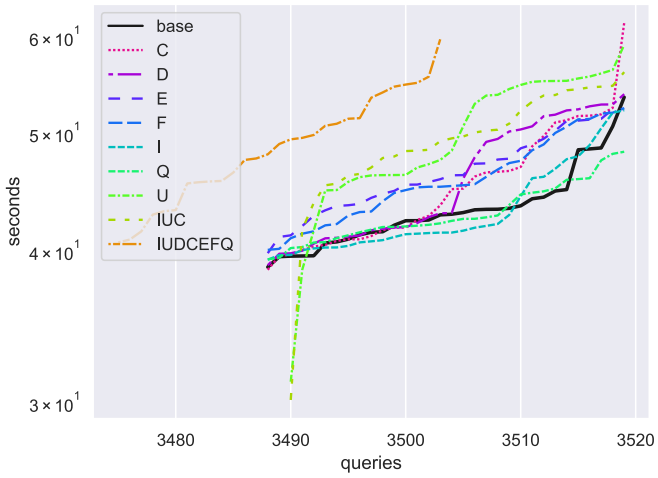
Figure 38: Reachability cardinality test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.



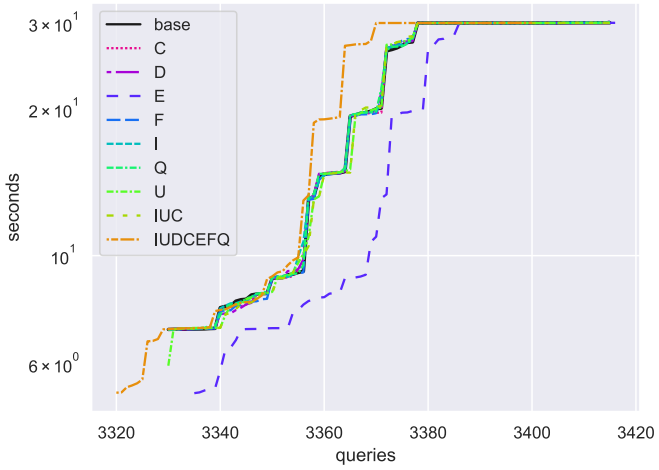
(a) Total time



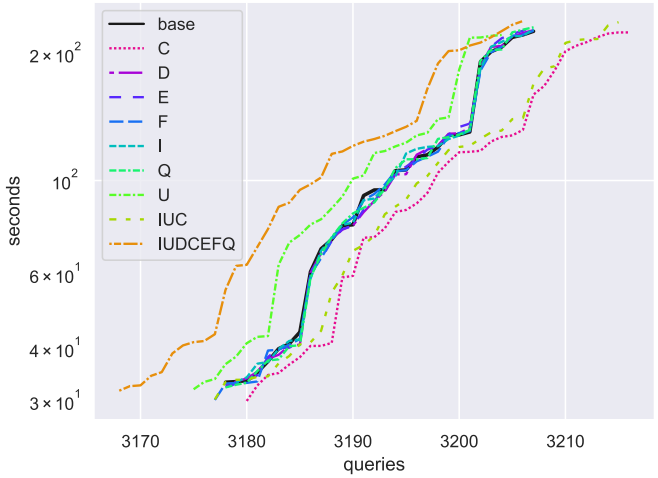
(b) State space size



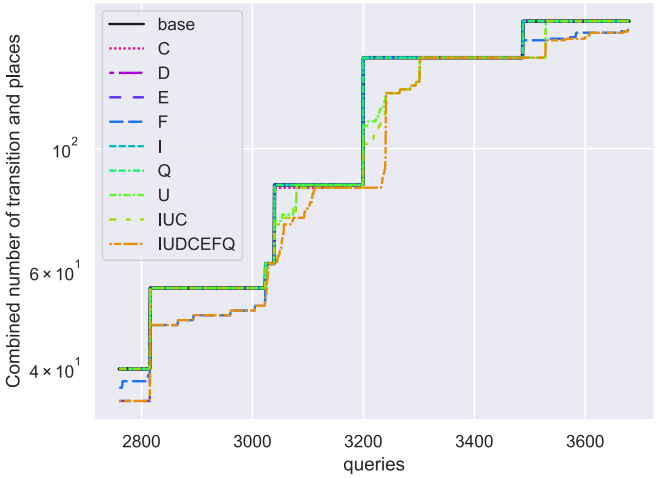
(c) Unfold time



(d) Reduce time

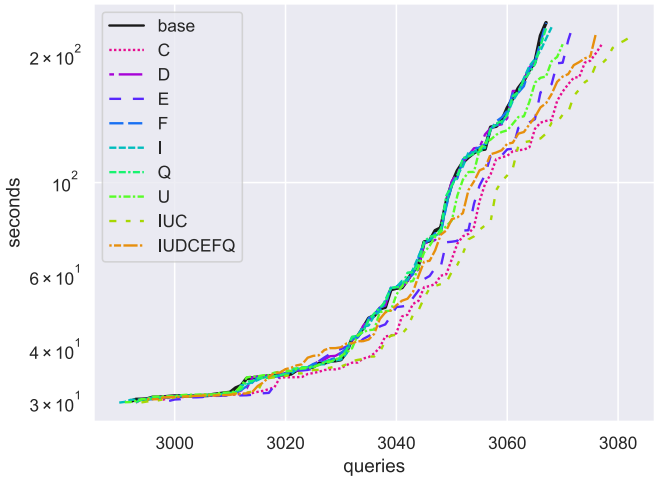


(e) Verification time

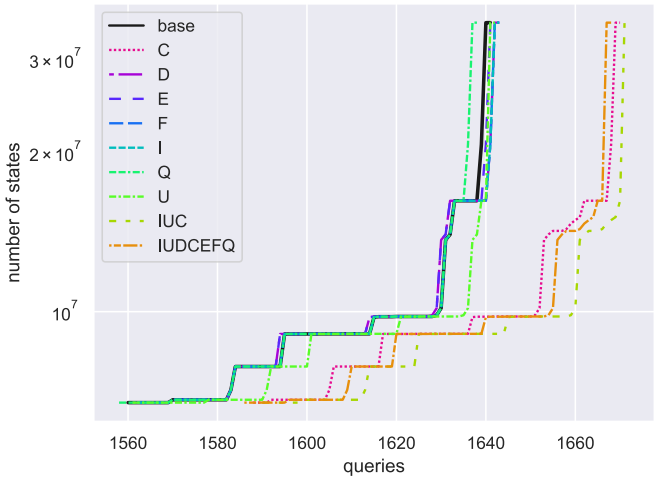


(f) CPN Reduced size

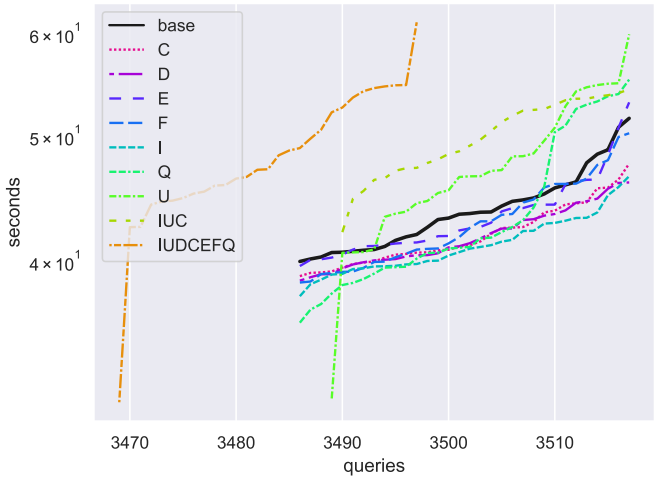
Figure 40: LTL cardinality test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.



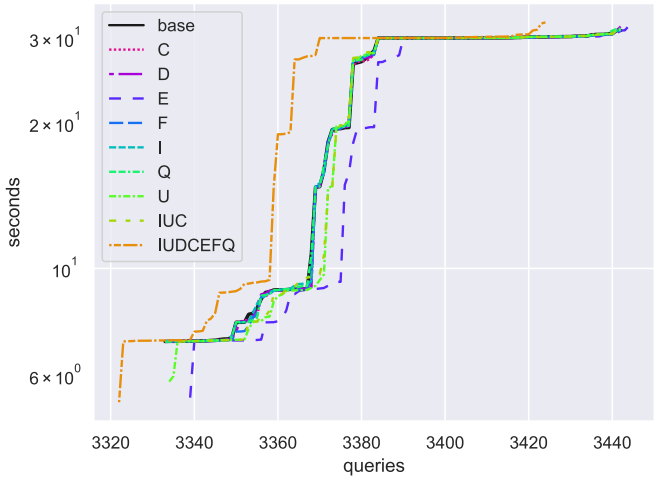
(a) Total time



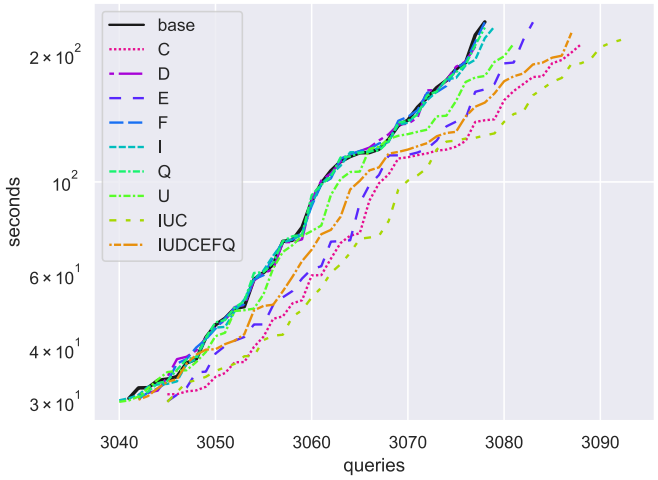
(b) State space size



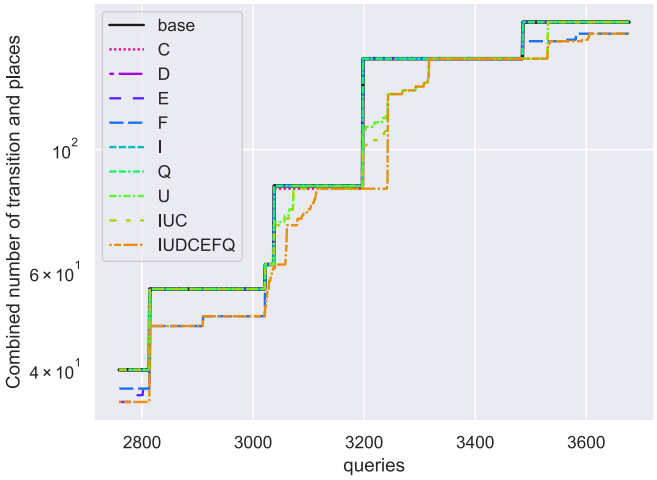
(c) Unfold time



(d) Reduce time

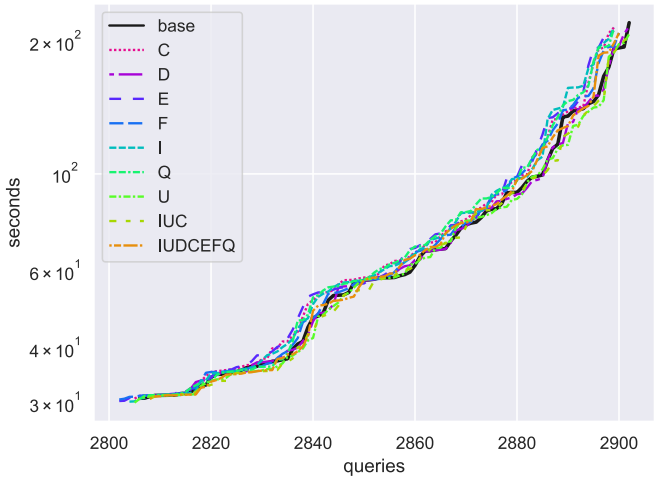


(e) Verification time

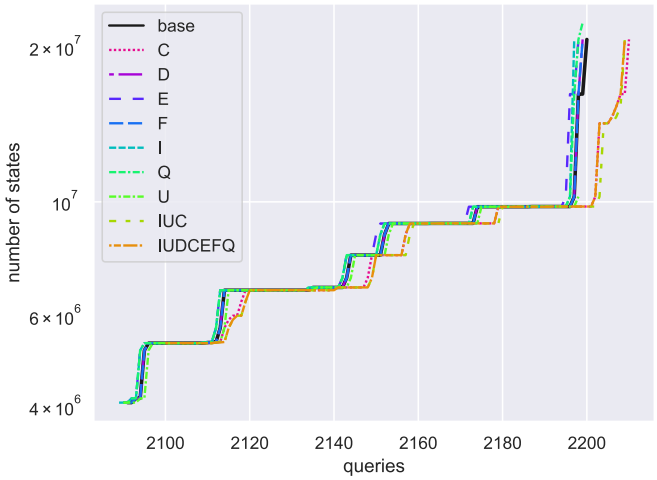


(f) CPN Reduced size

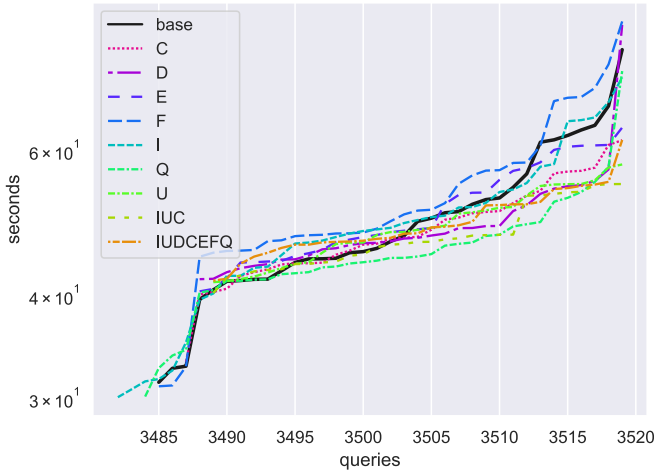
Figure 42: LTL fireability test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.



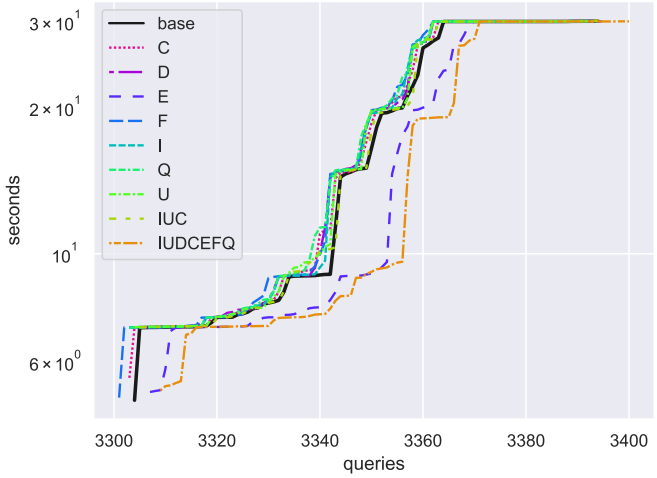
(a) Total time



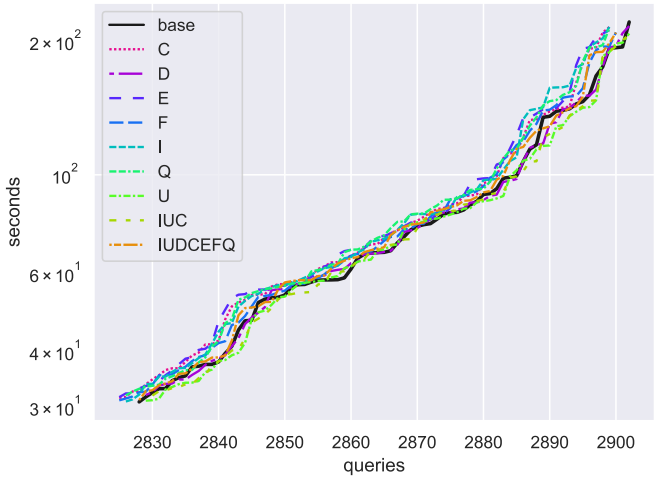
(b) State space size



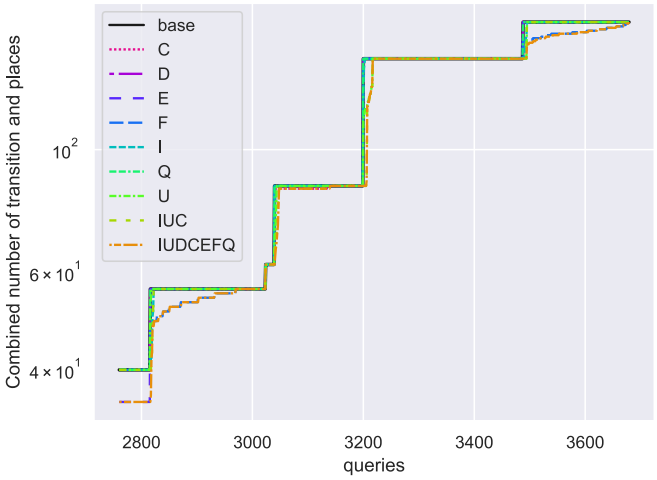
(c) Unfold time



(d) Reduce time

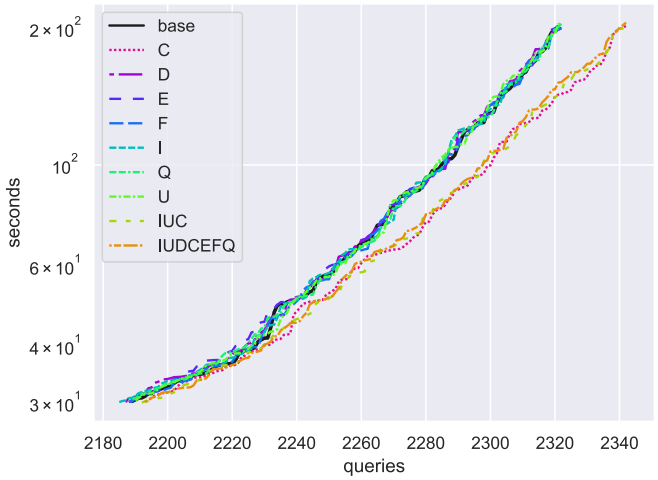


(e) Verification time

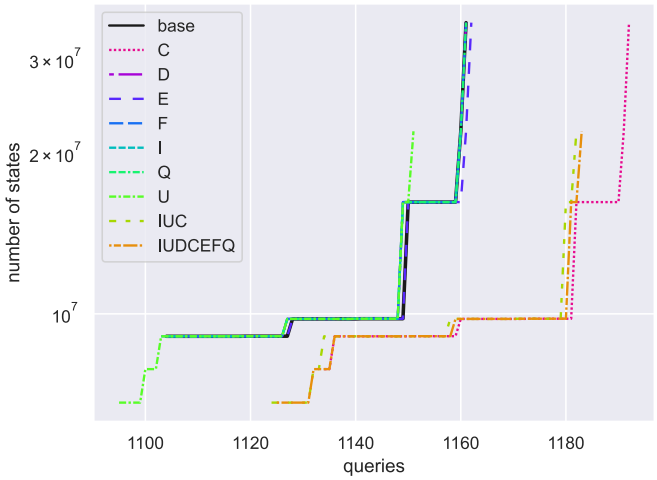


(f) CPN Reduced size

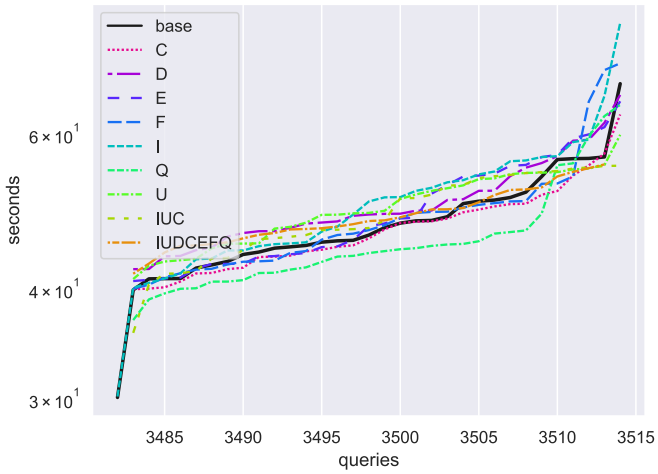
Figure 44: CTL cardinality test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.



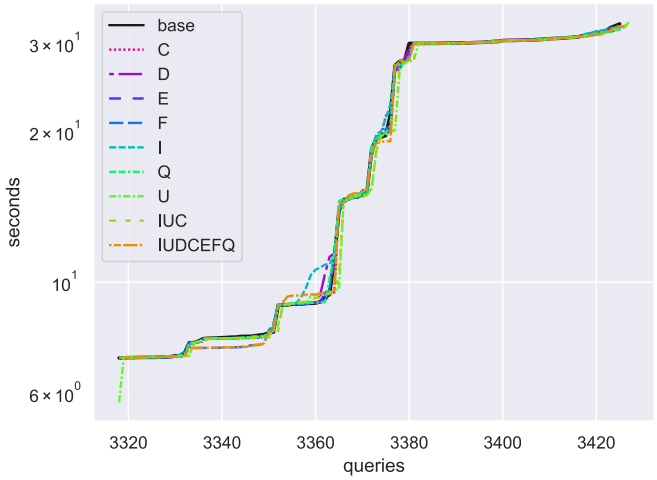
(a) Total time



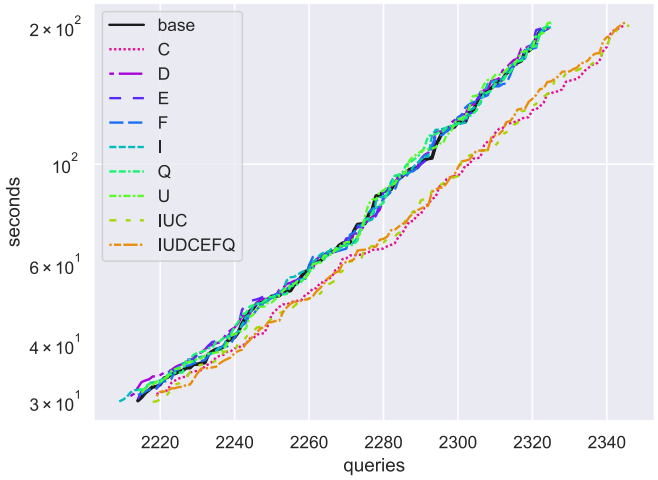
(b) State space size



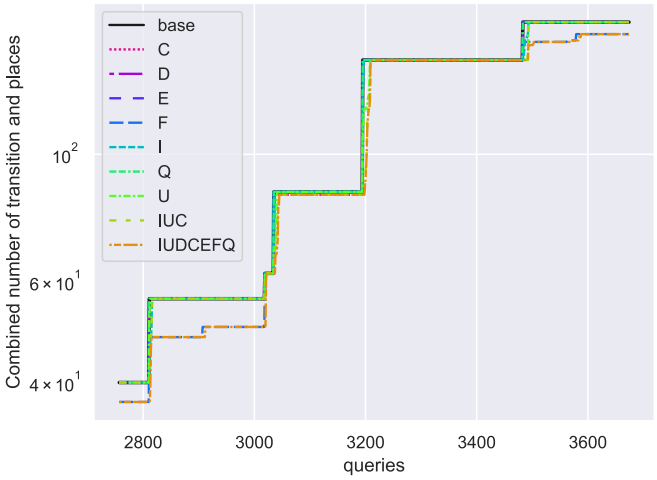
(c) Unfold time



(d) Reduce time

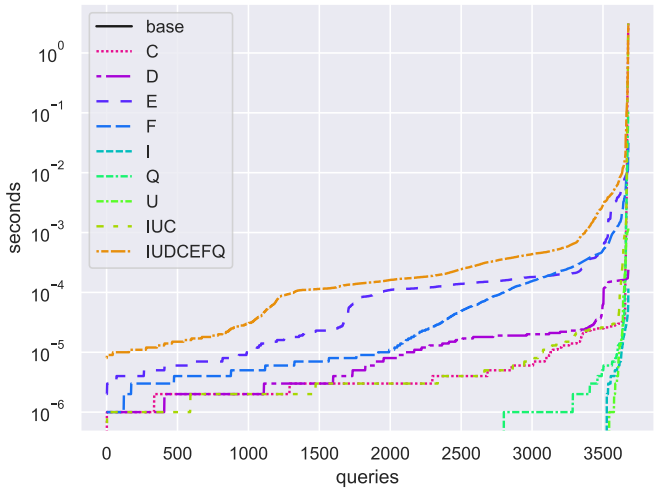


(e) Verification time

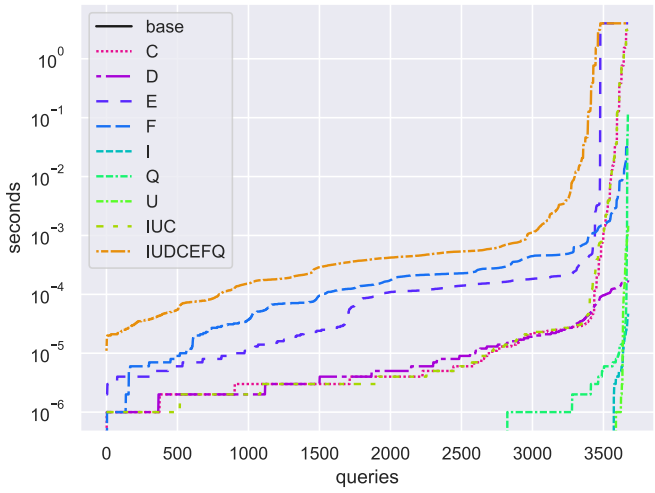


(f) CPN Reduced size

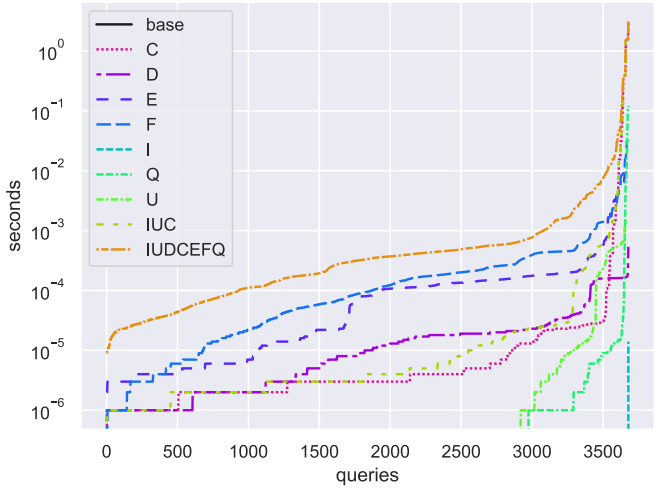
Figure 46: CTL fireability test results for CPN. (a) Total time spent (color reduction, unfolding, reduction and verification) for each experiment, > 30 sec. (b) Top 5% state space sizes found within timeout of reduced nets for each experiment. (c) Time spent in unfolding phase for each experiment, > 30 sec. (d) Time spent in reduction phase for each experiment, > 5 sec. (e) Time spent in verification phase for each experiment, > 30 sec. (f) Top 25% size (places+transitions) of Color reduced colored Petri nets. All data points are sorted by size.



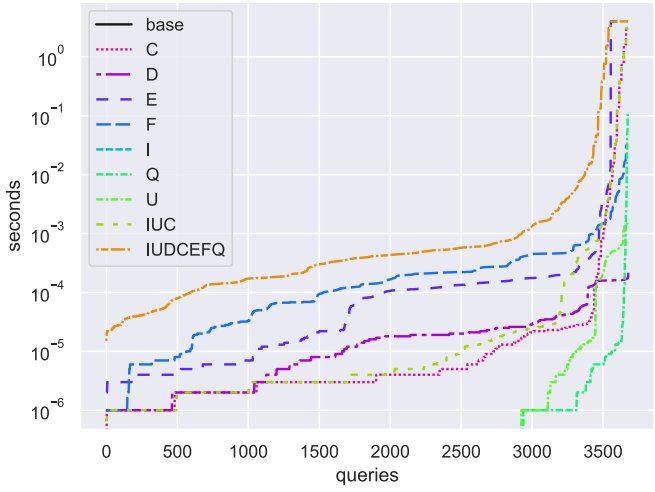
(a) CTLCardinality color reduction times



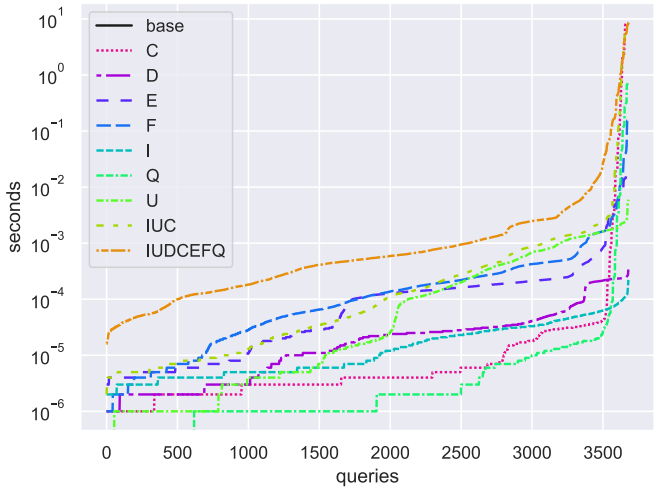
(b) CTLFireability color reduction times



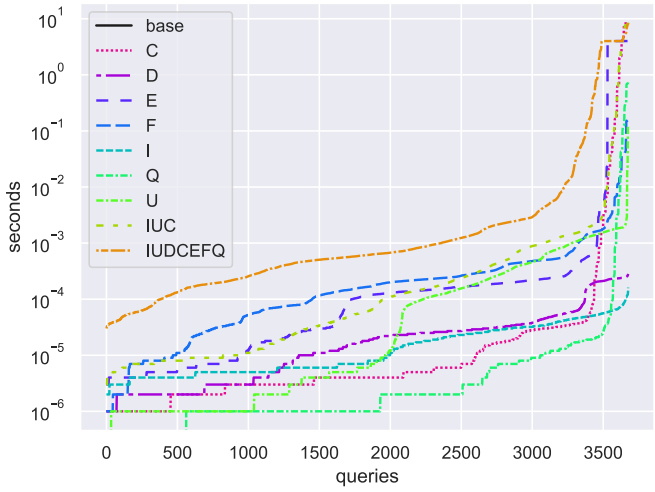
(c) LTLCardinality color reduction times



(d) LTLFireability color reduction times



(e) Reachability Cardinality color reduction times



(f) Reachability Fireability color reduction times

Figure 48: Color reduction times for experiments on in all categories

C.3 CPN answers by model type

In this appendix we have split the results shown in Section 6.8 Table 7 into each model type. This allows us to see if we uniformly gain answers no matter the model, or if there are specific models certain rules gain additional answers in. Table 8 shows additional answers by model in the Reachability categories. Table 9 shows additional answers by model in the CTL categories. Table 10 shows additional answers by model in the LTL categories.

model name	base	C	D	E	F	I	Q	U	IUC	IUDCEFQ
ReachabilityCardinality										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	128	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	297	-	-	-	-	-	-	-	-	-
CSRepetitions	90	-	-	-	-	-	-	-	-	-
DatabaseWithMutex	68	-	-	-	-	-	-	1	1	1
DrinkVendingMachine	112	-	-	-	-	-	-	-	-	-
FamilyReunion	77	-	-	-	-	2	-	-7	-2	-7
GlobalResAllocation	112	-	-	-	-	-	-	-	-	-
LamportFastMutEx	103	-	-	-	-	-	-	-	-	-
NeoElection	112	-	-	-	-	-	-	-	-	-
PermAdmissibility	96	-	-	-	-	-	-	-	-	-
Peterson	89	-	-	-	-	-	-	-	-	-
Philosophers	144	-	-	-	-	-	-1	1	1	1
PhilosophersDyn	67	-	-	-	-	-	-	-	-	-
PolyORBLF	263	-	-	-	-	-	-	3	3	3
PolyORBNT	105	21	-	-	-	-	-	-	21	21
QuasiCertifProtocol	109	-	-	-	-	-	-	-	-1	-1
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	107	-1	-1	-1	-	-	-1	-	-	-
SharedMemory	96	-	-	-	-	-	-	5	7	8
Sudoku	261	-	-	-	-	-	-	-	-	-
TokenRing	122	-	-2	-	-1	-	-2	-1	-1	-
UtilityControlRoom	270	-	-	-	-	-	-	-	-	-
VehicularWifi	13	-	-	-	-	-	-	-	-	-
ReachabilityFireability										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	126	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	221	-1	-	-1	-	-	-	-	-	-
CSRepetitions	70	-	-	-	-	1	-	-	1	1
DatabaseWithMutex	76	-	-	-	-	-	-	-	-	-
DrinkVendingMachine	112	-	-	-	-	-	-	-	-	-
FamilyReunion	76	-	-	-	-	1	-	-5	-3	-7
GlobalResAllocation	112	-	-	-	-	-	-	-	-	-
LamportFastMutEx	104	-	-	-	-	-	-	-	-	-
NeoElection	109	-	-	-	-	-	-	-	-	-
PermAdmissibility	96	-	-	-	-	-	-	-	-	-
Peterson	80	-	-	-	-	-	-	-	-	-
Philosophers	136	-	-	-	-	-	-	7	7	7
PhilosophersDyn	67	-1	-	-	-	-	-	-	-	-
PolyORBLF	193	1	-	1	-	1	1	1	1	1
PolyORBNT	57	64	1	1	-1	1	-	-2	56	58
QuasiCertifProtocol	100	-	-	-	-	4	-	2	5	5
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	96	2	1	-	-	-	1	1	2	3
SharedMemory	84	-	-	-	-	-	-	11	11	11
Sudoku	256	-	-	-	-	-	-	-	-	-
TokenRing	90	-	-	-	-	-	-	-	3	1
UtilityControlRoom	248	-	1	-	-	-	-	4	4	4
VehicularWifi	15	-	-	-	-	-	-	-	-	-

Table 8: CPN reachability additional answers for each model type

model name	base	C	D	E	F	I	Q	U	IUC	IUDCEFQ
CTLCardinality										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	126	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	245	-	-	-	-	-	-	-	-	-
CSRepetitions	88	-	-	-	-	-	-	-	-	-
DatabaseWithMutex	63	-	-	-	-	-	-	-	-	-
DrinkVendingMachine	109	-	-	-	-	-	-	-	-	-
FamilyReunion	39	-	-	-	-	-	-	-	-	-
GlobalResAllocation	106	-	-	-	-	-	-	-	-	-
LamportFastMutEx	98	-	-	-	-	-	-	-	-	-
NeoElection	101	-	-	-	-	-	-	-	-	-
PermAdmissibility	81	-	-	-	-	-	-	-	-	-
Peterson	85	-	-	-	-	-	-	-	-	-
Philosophers	133	-	-	-	-	-	-	-	-	-
PhilosophersDyn	69	-	1	-1	-	-1	-1	-	-	-
PolyORBLF	258	-	-	-	-	-	-	-1	-1	-1
PolyORBNT	144	-	-	-	-	-	-	-	-	-
QuasiCertifProtocol	82	-	-	-	-	-	-	-	-	-
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	90	-1	-1	-	-1	-	-	-	-	-
SharedMemory	88	-	-	-	-	-	-	1	1	1
Sudoku	267	-	-	-	-	-	-	-	-	-
TokenRing	118	-2	-	-2	-1	-2	-2	-	-	-2
UtilityControlRoom	231	-	-	-	-	-	-	-	-	-
VehicularWifi	10	-	-	-	-	-	-	-	-	-
CTLFireability										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	94	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	195	-	-	-	-	-	-	-	-	-1
CSRepetitions	61	-	-	-	-	-	-	-	-	-
DatabaseWithMutex	51	-	-	-	-	-	-	-	-	-
DrinkVendingMachine	104	-	-	-	-	-	-	-	-	-
FamilyReunion	36	-	-	-	-	-	-	-	-	-1
GlobalResAllocation	94	-	-	-	-	-	-	-	-	-
LamportFastMutEx	89	-	-	-	-	-	-	-	-	-
NeoElection	78	-	-	-	-	-	-	-	-	-
PermAdmissibility	76	-	-	-	-	-	-	-	-	-
Peterson	63	1	1	1	1	1	1	1	1	1
Philosophers	127	-	-	-	-	-	-	-1	-1	-
PhilosophersDyn	67	-	-	-	-	-	-	-	-	-
PolyORBLF	187	-	-	-	-	-	-	-	-	-
PolyORBNT	91	21	-	-	-	-	-	-	21	21
QuasiCertifProtocol	75	-	-	-	-	-	-	-	-	-
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	64	-	-	-	-	-	-	-	-	1
SharedMemory	53	-1	-	-1	-	-	-	1	1	-
Sudoku	180	-	-	-	-	-	-	-	-	-
TokenRing	73	-	-	-	-	-	-	-	-	-
UtilityControlRoom	180	-	-	-	-	-	-	-	-	-
VehicularWifi	12	-	-	-	-	-	-	-	-	-

Table 9: CPN CTL additional answers for each model type

model name	base	C	D	E	F	I	Q	U	IUC	IUDCEFQ
LTL Cardinality										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	128	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	295	-	-	-	-	-	-	-1	-1	-1
CSRepetitions	87	-	-	-	-	-	-	-	-	-
DatabaseWithMutex	69	-	-	-	-	-	-	-	-	-
DrinkVendingMachine	111	-	-	-	-	-	-	-	-	-
FamilyReunion	62	-	-	-	-	-	-	-	-	-10
GlobalResAllocation	111	-	-	-	-	-	-	-	-	-
LamportFastMutEx	111	-	-	-	-	-	-	-	-	-
NeoElection	110	-	-	-	-	-	-	-	-	-
PermAdmissibility	90	-	-	-	-	-	-	-	-	-
Peterson	94	-	-	-	-	-	-	-	-	-
Philosophers	176	-	-	-	-	-	-	-	-	-
PhilosophersDyn	75	-	-	-	-	-	-	-	-	-
PolyORBLF	269	-	-	-	-	-	-	-	-	-
PolyORBNT	145	9	-	-	-	-	-	-2	9	9
QuasiCertifProtocol	96	-	-	-	-	-	-	-	-	-
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	109	-	-	-	-	-	-	-	-	-
SharedMemory	106	-	-	-	-	-	-	-	-	-
Sudoku	289	-	-	-	-	-	-	-	-	-
TokenRing	124	-	-	-	-	-	-	-	-	-
UtilityControlRoom	261	-	-	-	-	-	-	1	1	2
VehicularWifi	15	-	-	-	-	-	-	-	-	-
LTLFireability										
AirplaneLD	144	-	-	-	-	-	-	-	-	-
BART	119	-	-	-	-	-	-	-	-	-
BridgeAndVehicles	275	-	-	-	-	-	-	-	-	-
CSRepetitions	91	-	-	-	-	-	-	-	-	-
DatabaseWithMutex	69	-	-	-	-	-	-	1	1	1
DrinkVendingMachine	110	-	-	-	-	-	-	-	-	-
FamilyReunion	60	-	-	-	-	-	-	1	2	-9
GlobalResAllocation	109	-	-	-	-	-	-	-	-	-
LamportFastMutEx	107	-	-	-	-	-	-	-	-	-
NeoElection	104	-	-	-	-	-	-	-	-	-
PermAdmissibility	90	-	-	-	-	-	-	-	-	-
Peterson	95	-	-	-	-	-	-	-	-	-
Philosophers	181	-	-	-	-	-	-	-	-	-
PhilosophersDyn	72	-	-	-	-	-	-	-	-	-
PolyORBLF	255	-	-	-	-	-	-	1	1	1
PolyORBNT	129	10	-	-	-	-	-	-1	10	10
QuasiCertifProtocol	95	-	-1	-	-	-	-	-1	-1	-1
Referendum	128	-	-	-	-	-	-	-	-	-
SafeBus	98	-	-	-	-	1	-	-	-	-
SharedMemory	108	-	-	-	-	-	-	-	-	-
Sudoku	269	-	-	-	-	-	-	-	-	-
TokenRing	98	-	-	5	-	-	-	-	-	5
UtilityControlRoom	247	-	-	-	-	-	-	2	2	2
VehicularWifi	15	-	-	-	-	-	-	-	-	-

Table 10: CPN LTL additional answers for each model type

C.4 P/T answers by model type

In this appendix we have split the results shown in Section 5.3 Table 4 into each model type. As there are many more model types for P/T nets than what we can reasonably fit into the thesis, rows which contains no additional answers are left out. Tables are sorted first by number of additional answers by the *base+CS* column, and then *base+C*. This allows us to see if we uniformly gain answers no matter the model, or if there are specific models certain rules gain additional answers in. Table 11 shows additional answers by model in the Reachability categories. Table 12 shows additional answers by model in the CTL categories. Table 13 shows additional answers by model in the LTL categories.

model name	base	base+C	base+CS
ReachabilityCardinality			
DLCround	315	0	37
MAPK	140	0	25
DLCflexbar	202	0	22
DLCshifumi	138	0	22
CloudDeployment	168	0	21
NoC3x3	217	0	12
FlexibleBarrier	310	0	10
MultiCrashLeafsetExtension	347	2	2
CSRepetitions	91	0	2
Kanban	205	0	2
RobotManipulation	196	0	2
SemanticWebServices	234	0	2
UtilityControlRoom	269	0	2
TwoPhaseLocking	346	-1	2
ClientsAndServers	317	0	1
DoubleExponent	97	0	1
EnergyBus	13	0	1
GPUForwardProgress	316	0	1
ShieldIIPt	290	0	1
ShieldRVt	335	0	1
RERS17pb114	31	-1	0
AirplaneLD	144	0	-1
PolyORBLF	260	0	-1
RERS17pb113	78	0	-1
Raft	143	0	-1
RERS17pb115	35	-1	-1
DatabaseWithMutex	79	0	-2
PermAdmissibility	91	0	-2
HealthRecord	271	0	-4
SharedMemory	91	-2	-4
ASLink	223	0	-8
ReachabilityFireability			
DLCround	279	0	73
DLCflexbar	175	0	49
DLCshifumi	136	0	24
FlexibleBarrier	296	-1	22
NoC3x3	176	0	14
UtilityControlRoom	246	0	14
MultiCrashLeafsetExtension	272	0	11
CloudDeployment	182	-1	10
ShieldRVt	309	-1	6
CSRepetitions	78	0	4
FamilyReunion	72	0	4
SafeBus	66	-1	4
CANConstruction	154	0	3
PolyORBNT	85	1	2
SharedMemory	78	0	2
ShieldPPPt	331	0	2
SimpleLoadBal	70	0	2
DNAwalker	199	1	1
RERS17pb113	30	1	1
SmallOperatingSystem	296	1	1
CloudReconfiguration	351	0	1
DES	310	0	1
MultiwaySync	13	0	1
PolyORBLF	238	0	1
RERS17pb115	2	-1	0
Raft	134	0	-1
HealthRecord	270	0	-2
AirplaneLD	143	0	-7
ASLink	135	0	-15

Table 11: P/T Reachability additional answers for each model type

model name	base	base+C	base+CS
CTLCardinality			
MAPK	67	1	2
DLCround	293	0	2
MultiCrashLeafsetExtension	375	1	1
DLCflexbar	188	0	1
NoC3x3	208	0	1
RERS17pb113	93	0	1
TCPcondis	90	0	1
Kanban	102	0	-1
PolyORBLF	262	0	-1
SquareGrid	44	0	-1
CloudReconfiguration	328	-1	-1
CTLFireability			
DLCshifumi	72	0	2
NoC3x3	138	0	2
Solitaire	44	3	1
GlobalResAllocation	23	1	1
BART	79	0	1
CloudDeployment	145	0	1
DLCflexbar	123	0	1
ClientsAndServers	211	0	-1
PolyORBNT	95	0	-1
SafeBus	54	0	-1
ASLink	201	0	-2

Table 12: P/T CTL additional answers for each model type

model name	base	base+C	base+CS
LTLCardinality			
MultiCrashLeafsetExtension	407	0	2
HouseConstruction	191	-1	2
CloudDeployment	173	0	1
DLCround	340	0	1
MAPK	164	0	1
RERS17pb115	114	0	1
RobotManipulation	205	0	1
LTLFireability			
DLCflexbar	196	0	3
SwimmingPool	150	0	2
HouseConstruction	200	0	1
CloudReconfiguration	328	-1	1
BridgeAndVehicles	280	0	-1
ERK	93	0	-1
FlexibleBarrier	290	0	-1
SmallOperatingSystem	300	-1	-1

Table 13: P/T LTL additional answers for each model type

D Canonical form and term unification

In CPNs, some guards may be syntactically different, but equivalent semantically, e.g. let $x, y \in \mathbf{Var}_C$ then $x = y$ and $x \leq y \wedge x \geq y$ are equivalent. For structural reductions, it is beneficial to know if two guards are semantically equivalent. Hence, we want to rewrite such guards to a canonical form, making it easier to compare guards. In this rewrite we can also consider term unification. E.g. if $x = y$ according to the guard, then we can replace all occurances of y with x in the guard and on the associated arcs.

In short, our guards consist of disjunctions, conjunctions, negations, and comparisons. In short, our guards consist of disjunctions, conjunctions, negations, and comparisons. Hence, a disjunctive normal form (DNF) is fitting. We can also eliminate all negations, since comparisons can be negated. To achieve syntactical equivalence between semantically equivalent guards, we must also define an order in which conjunction clauses and the comparisons in the conjunctions appear. Assume that $(\mathbf{Var}, \preceq)$ is a total order of all variables. This order induces a lexicographical order of comparisons, which induces a lexicographical ordering of conjunctions in the DNF. From now on assume all guards are written lexicographically. To construct the canonical form, we must also simplify, reduce, and combine as many comparisons as possible. Another thing to note is, that the term unification algorithms, such as Martelli/Montanari's algorithm [22], operate on a set of equalities (i.e. a single conjunction), while we have disjunctions and more types of comparisons. Therefore, term unification is only possible, when every conjunction clause of the DNF contain the comparison $x = \tau_C$, in which case we can substitute every x with τ_C.

With all of the above in mind, a potential algorithm to construct a canonical form of transition t with guard $G(t)$ follows:

1. Let $\sigma : \mathbf{Var}_C \to C$ be a substitution/unification mapping, with initially $\sigma := \{\}$.
2. Reduce color expressions by noticing that $\tau_C\texttt{++}$ and $\tau_C\texttt{--}$ are dual functions with one parameter. Hence we have that:

$$\tau_C\texttt{++}\texttt{--} \Rightarrow \tau_C$$
$$\tau_C\texttt{--}\texttt{++} \Rightarrow \tau_C$$

3. Decompose comparisons when both sides are a successor or predecessor expressions:

$$\tau_C^1\texttt{++} \bowtie \tau_C^2\texttt{++} \Rightarrow \tau_C^1 \bowtie \tau_C^2$$
$$\tau_C^1\texttt{--} \bowtie \tau_C^2\texttt{--} \Rightarrow \tau_C^1 \bowtie \tau_C^2$$

4. Prefer successor over predecessor:

$$\tau_C^1\texttt{--} \bowtie \tau_C^2 \Rightarrow \tau_C^1 \bowtie \tau_C^2\texttt{++}$$
$$\tau_C^1 \bowtie \tau_C^2\texttt{--} \Rightarrow \tau_C^1\texttt{++} \bowtie \tau_C^2$$

5. Push all negations to the bottom using De Morgan's law and double negation elimination:

$$\neg(\gamma_1 \wedge \gamma_2) \Rightarrow \neg\gamma_1 \vee \neg\gamma_2$$
$$\neg(\gamma_1 \vee \gamma_2) \Rightarrow \neg\gamma_1 \wedge \neg\gamma_2$$
$$\neg\neg\gamma \Rightarrow \gamma$$

6. To eliminate the remaining negations, we negate comparisons

$$\neg(\tau_C^1 < \tau_C^2) \Rightarrow \tau_C^1 \geq \tau_C^2$$
$$\neg(\tau_C^1 \leq \tau_C^2) \Rightarrow \tau_C^1 > \tau_C^2$$
$$\neg(\tau_C^1 = \tau_C^2) \Rightarrow \tau_C^1 \neq \tau_C^2$$
$$\neg(\tau_C^1 \geq \tau_C^2) \Rightarrow \tau_C^1 < \tau_C^2$$
$$\neg(\tau_C^1 > \tau_C^2) \Rightarrow \tau_C^1 \leq \tau_C^2$$
$$\neg(\tau_C^1 \neq \tau_C^2) \Rightarrow \tau_C^1 = \tau_C^2$$

7. Consider $\bowtie \in \{<, \leq, =, \geq, >, \neq\}$. We can flip some of these comparisons, reducing $\bowtie$ to $\{<, \leq, =, \neq\}$

$$\tau_C^1 < \tau_C^2 \Rightarrow \tau_C^1 < \tau_C^2$$
$$\tau_C^1 \leq \tau_C^2 \Rightarrow \tau_C^1 \leq \tau_C^2$$
$$\tau_C^1 = \tau_C^2 \Rightarrow \tau_C^1 = \tau_C^2$$
$$\tau_C^1 \geq \tau_C^2 \Rightarrow \tau_C^2 \leq \tau_C^1$$
$$\tau_C^1 > \tau_C^2 \Rightarrow \tau_C^2 < \tau_C^1$$
$$\tau_C^1 \neq \tau_C^2 \Rightarrow \tau_C^1 \neq \tau_C^2$$

8. Rewrite to minimal disjunctive normal form (DNF) using the Quine–McCluskey algorithm. Each comparison $\tau_C^1 \bowtie \tau_C^2$ is considered a boolean variable during this algorithm, and syntactically equivalent comparisons are considered to be the same boolean variable.

9. Let $c_1, c_2 \in C$ and reduce and combine comparisons by applying the following equivalences:

$$
\begin{aligned}
&\gamma \wedge \gamma \Rightarrow \gamma && \\
&\gamma \vee \gamma \Rightarrow \gamma && \\
&\top \vee \gamma \Rightarrow \top && \\
&\bot \vee \gamma \Rightarrow \gamma && \\
&\top \wedge \gamma \Rightarrow \gamma && \\
&\bot \wedge \gamma \Rightarrow \bot && \\
&c_1 < c_2 \Rightarrow \top && \textbf{if } c_1 < c_2 \\
&c_1 < c_2 \Rightarrow \bot && \textbf{if } c_1 \geq c_2 \\
&c_1 \leq c_2 \Rightarrow \top && \textbf{if } c_1 \leq c_2 \\
&c_1 \leq c_2 \Rightarrow \bot && \textbf{if } c_1 > c_2 \\
&c_1 = c_2 \Rightarrow \bot && \textbf{if } c_1 \neq c_2 \\
&c_1 = c_2 \Rightarrow \top && \textbf{if } c_1 = c_2 \\
&\tau_C = \tau_C \Rightarrow \top && \\
&\tau_C^1 < \tau_C^2 \vee \tau_C^1 = \tau_C^2 \Rightarrow \tau_C^1 \leq \tau_C^2 && \\
&\tau_C^1 < \tau_C^2 \vee \tau_C^2 < \tau_C^1 \Rightarrow \tau_C^1 \neq \tau_C^2 && \\
&\tau_C^1 < \tau_C^2 \vee \tau_C^2 \leq \tau_C^1 \Rightarrow \top && \\
&\tau_C^1 < \tau_C^2 \wedge \tau_C^1 \leq \tau_C^2 \Rightarrow \tau_C^1 < \tau_C^2 && \\
&\tau_C^1 < \tau_C^2 \vee \tau_C^1 \leq \tau_C^2 \Rightarrow \tau_C^1 \leq \tau_C^2 && \\
&\tau_C^1 < \tau_C^2 \wedge \tau_C^1 = \tau_C^2 \Rightarrow \bot && \\
&\tau_C^1 < \tau_C^2 \wedge \tau_C^2 < \tau_C^1 \Rightarrow \bot && \\
&\tau_C^1 \leq \tau_C^2 \wedge \tau_C^1 \neq \tau_C^2 \Rightarrow \tau_C^1 < \tau_C^2 && \\
&\tau_C^1 \leq \tau_C^2 \wedge \tau_C^2 \leq \tau_C^1 \Rightarrow \tau_C^1 = \tau_C^2 && \\
&\tau_C^1 = \tau_C^2 \wedge \tau_C^2 \neq \tau_C^1 \Rightarrow \bot &&
\end{aligned}
$$

10. If $x = \tau_C$ appears in every conjunction clause of the DNF, then update $\sigma := \sigma \cup \{x \mapsto \tau_C\}$ and then replace every occurrance of x with τ_C.
    If at least one reduction was done in step 9, go back to 8.
11. We cannot have substitute variables with concrete colors on the arcs, so we have to re-insert those in the guard. For every $(x \mapsto \tau_C) \in \sigma$ do:
    - If $\tau_C \neq y$, then insert the comparison $x = \tau_C$ in every conjunctive clause of the DNF.
    - If $\tau_C = y$, then for all $p \in P$ update $\boxminus(p,t)$ to $\boxminus'(p,t)$ such that $\boxminus'(p,t)(y) := \boxminus(p,t)(y) + \boxminus(p,t)(x)$ and $\boxminus'(p,t)(x) := 0$. Similarly, for $\boxplus(t,p)$.
12. For every color type $C \in \mathbb{D}$ let $\mathbf{Var}_C = \{x_1, x_2, \ldots, x_n\}$ such that $x_1 \prec x_2 \prec \cdots \prec x_n$ and let $\mathbf{Vars}(t) \cap \mathbf{Var}_C = \{y_1, y_2, , y_m\}$ such that $y_1 \prec y_2 \prec \cdots \prec y_n$. For each integer $i \in [1, m]$ starting for 1, replace all occurrences of y_i

with x_i in $G(t)$ and on every arc's multiset, $\boxminus(p,t)$ and $\boxplus(t,p)$. This way, the transition now uses the smallest variables while lexicographical order is maintained.

In the CPNs in MCC2021, we found no transitions for which the normal form allows us to do additional reductions (so far).