Improvements in Unfolding of Colored Petri Nets

Alexander Bilgram, Peter G. Jensen, Thomas Pedersen,
Jiří Srba, and Peter H. Taankvist

Aalborg University
Department of Computer Science, Aalborg, Denmark

Abstract. Colored Petri nets offer a compact and user friendly representation of the traditional P/T nets and colored nets with finite color ranges can be unfolded into the underlying P/T nets, however, at the expense of an exponential explosion in size. We present two novel techniques based on static analyses in order to reduce the size of unfolded colored nets. The first method identifies colors that behave equivalently and groups them into equivalence classes, potentially reducing the number of used colors. The second method overapproximates the sets of colors that can appear in places and excludes colors that can never be present in a given place. Both methods are complementary and the combined approach allows us to significantly reduce the size of multiple colored Petri nets from the Model Checking Contest benchmark. We compare the performance of our unfolder with state-of-the-art techniques implemented in the tools MCC, Spike and ITS-Tools, and while our approach remains competitive w.r.t. unfolding time, it outperforms the existing approaches both in the size of unfolded nets as well as in the number of answered model checking queries from the 2020 Model Checking Contest.

1 Introduction

Petri nets [22], also known as P/T nets, are a powerful modelling formalism supported by a rich family of verification techniques [20]. However, P/T nets often become too large and incomprehensible for humans to read. Therefore, colored Petri nets (CPN) [14] were introduced to allow for high level modelling of distributed systems. In CPNs, each place is assigned a color domain and each token in that place has a color from its domain. Arcs have expressions that define what colored tokens to consume or produce, and transitions have guard expressions that restrict transition enabledness.

A CPN can be translated into an equivalent P/T net, provided that every color domain is finite, through a process called *unfolding*. This allows us to use efficient verification tools already developed for P/T nets. When unfolding a CPN, each place is unfolded into a new place for each color that a token can take in that place; a naive approach is to create a new place for each color in the color domain of the place. Transitions are unfolded such that each binding of variables to colors, satisfying the guard, is unfolded into a new transition copy in the unfolded net. The size of an unfolded net can be exponentially larger

than the colored net and the unfolding process therefore requires optimizations in order to finish in realistic time and memory. Several types of improvements were proposed that analyse transition guards and arc expressions [6, 19, 23]. However, even with these optimizations, there still exist CPNs that cannot be unfolded. As an example, the largest instances of the nets *FamilyReunion* [12, 5] and *DrinkVendingMachine* [11, 21] from the Model Checking Contest [18] have not been unfolded yet.

We propose two novel methods for statically analysing a CPN to reduce the size of the unfolded P/T net. The first method called *color quotienting* uses the fact that sometimes multiple colors behave equivalently throughout the colored net. If such colors exist in the net, we can create equivalence classes that represent the colors with similar behaviour. As such, we can reduce the amount of colors that we need to consider when unfolding. The second method called *color approximation* overapproximates which colors can possibly be present in any given place s.t. we only unfold places for the colors that can exist. This method also allows for invalidating bindings that are dependent on unreachable colors, thus reducing the amount of transitions that are unfolded.

Our two methods are implemented in the model checker TAPAAL [7, 13] and an extensive experimental evaluation shows convincing performance compared to the state-of-the-art tools for CPN unfolding.

Related work. Heiner et al. [19] analyse the arc and guard expressions to reduce the amount of bindings by collecting *patterns*. The pattern analysis is implemented in the tool Snoopy [9] and our color approximation method further extends this method. In [23] the same authors present a technique for representing the patterns as Interval Decision Diagrams. This technique is used in the tools Snoopy [9], MARCIE [10] and Spike [3] and performs better compared to [19]; it also allows to unfold a superset of colored nets compared to the format adopted by the Model Checking Contest benchmark [18].

In [6] (MCC) Dal-Zilio describes a method called *stable places*. A stable place is a place that never changes from the initial marking, i.e. every time a token is consumed from this place an equivalent token is added to the place. This method is especially efficient on the net BART from the Model Checking Contest [18], however, it does not detect places that deviate even a little from the initial marking. Our color approximation method includes a more general form of the stable places. In the unfolder MCC [6], a *component analysis* is introduced and it detects if a net consists of a number of copies of the same component. MCC is used in the TINA toolchain [1] and to our knowledge in the latest release of the LoLA tool [27]. GreatSPN [8] is another tool for unfolding CPNs, however, in [6] it is demonstrated that MCC is able to greatly outperform GreatSPN and as such we omit GreatSPN from later experiments.

ITS-Tools [24] has an integrated unfolding engine. The tool uses a technique of *variable symmetry identification*, in which it is analyzed whether variables x and y are permutable in a binding. Furthermore, they use stable places during the binding and they apply analysis to choose the binding order of parameters to simplify false guards as soon as possible. After unfolding, ITS-Tools applies

further post-unfolding reductions that remove orphan places/transitions and behaviourally equivalent transitions [25]. Our implementation includes a variant of the symmetric variables reduction as well. In [26] Thierry-Mieg et al. present a technique for automatic detection of symmetries in high level Petri nets used to construct symbolic reachability graphs in the GreatSPN tool. This detection of symmetries is reminiscent of the color quotienting method presented in this paper, although our color quotienting method is used for unfolding the colored Petri net instead of symbolic model checking.

In [17] Klostergaard presents a simple unfolding method implemented in TAPAAL [7, 13], which is the base of our implementation. The implementation is efficient but there are several nets which it cannot unfold. Both unfolding methods introduced in this paper are advanced static analyses techniques and all of the above mentioned techniques, except symmetric variables and component analysis, are captured by color approximation and/or color quotienting.

2 Preliminaries

Let $\mathbb{N}^{>0}$ be the set of positive integers and $\mathbb{N}^0$ the set of nonnegative integers. A Labeled Transition System (LTS) is a triple $(Q, Act, \rightarrow)$ where Q is a set of states, Act is a finite, nonempty set of actions, and $\rightarrow \subseteq Q \times Act \times Q$ is the transition relation. A binary relation R over the set of states of an LTS is a *bisimulation* iff for every $(s_1, s_2) \in R$ and $a \in Act$ it holds that if $s_1 \xrightarrow{a} s_1'$ then there is a transition $s_2 \xrightarrow{a} s_2'$ s.t. $(s_1', s_2') \in R$, and if $s_2 \xrightarrow{a} s_2'$ then there is a transition $s_1 \xrightarrow{a} s_1'$ s.t. $(s_1', s_2') \in R$. Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation R s.t. $(s, s') \in R$.

A finite *multiset* over some nonempty set A is a collection of elements from A where each element occurs in the multiset a finite amount of times; a multiset S over a set A can be identified with a function $S : A \rightarrow \mathbb{N}^0$ where $S(a)$ is the number of occurrences of element $a \in A$ in the multiset S. We shall represent multisets by a formal sum $\sum_{a \in A} S(a)'(a)$ such that e.g. $1'(x) + 2'(y)$ stands for a multiset containing one element x and two elements y. We assume the standard multiset operations of membership ($\in$), inclusion ($\subseteq$), equality ($=$), union ($\uplus$), subtraction ($\setminus$) and by $|S|$ we denote the cardinality of S (including the repetition of elements). By $\mathcal{S}(A)$ we denote the set of all multisets over the set A. Finally, we also define the function *set* as a way of reducing multisets of colors to sets of colors given by $set(S) \stackrel{\text{def}}{=} \{a \mid a \in S\}$ where $set(S)$ is the set of all colors with at least one occurrence in S.

2.1 Colored Petri Nets

Colored Petri nets (CPN) are an extension of traditional P/T nets introduced by Kurt Jensen [14] in 1981. In CPNs, places are associated with color domains where colors represent the values of tokens. Arc expressions describe what colors to consume and add to places depending on a given binding (assignment of variables to colors). Transitions may contain guards restricting which bindings are

valid. There exist several different definitions of CPNs from the powerful version defined in [16] that includes the ML language for describing arcs expressions and guards to less powerful ones such as the one used in the Model Checking Contest [18]. We shall first give an abstract definition of a CPN.

Definition 1. *A colored Petri net is a tuple* $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ *where*

1. P *is a finite set of places,*
2. T *is a finite set of transitions s.t.* $P \cap T = \emptyset$*,*
3. $\mathbb{C}$ *is a nonempty set of colors,*
4. $\mathbb{B}$ *is a nonempty set of bindings,*
5. $C : P \to 2^{\mathbb{C}} \setminus \emptyset$ *is a place color type function,*
6. $G : T \times \mathbb{B} \to \{true, false\}$ *is a guard evaluation function,*
7. $W : ((P \times T) \cup (T \times P)) \times \mathbb{B} \to \mathcal{S}(\mathbb{C})$ *is an arc evaluation function s.t.* $set(W((p,t),b)) \subseteq C(p)$ *and* $set(W((t,p),b)) \subseteq C(p)$ *for all* $p \in P$*,* $t \in T$ *and* $b \in \mathbb{B}$*,*
8. $W_I : P \times T \to \mathbb{N}^{>0} \cup \{\infty\}$ *is an inhibitor arc weight function, and*
9. M_0 *is the initial marking where a marking* M *is a function* $M : P \to \mathcal{S}(\mathbb{C})$ *s.t.* $set(M(p)) \subseteq C(p)$ *for all* $p \in P$*.*

Notice that G, W and W_I are semantic functions which are in different variants of CPN defined by a concrete syntax. The set of all markings on a CPN $\mathcal{N}$ is denoted by $\mathbb{M}(\mathcal{N})$. In order to avoid the use of partial functions, we allow $W((p,t),b) = W((t,p),b) = \emptyset$ and $W_I(p,t) = \infty$, meaning that if the arc evaluation function returns the empty multiset then the arc has no effect on transition firing and if the inhibitor arc function returns infinity then it never inhibits the connected transition.

Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a fixed CPN for the rest of this section. Let $B(t) \stackrel{\text{def}}{=} \{b \in \mathbb{B} \mid G(t,b) = true\}$ be the set of all bindings that satisfy the guard of transition $t \in T$. Let $\ell : T \to Act$ be a transition labeling function. The semantics of a CPN $\mathcal{N}$ is defined as an LTS $L(\mathcal{N}) = (\mathbb{M}(\mathcal{N}), Act, \to)$ where $\mathbb{M}(\mathcal{N})$ is the set of states defined as all markings on $\mathcal{N}$, Act is the set of actions, and $M \xrightarrow{a} M'$ iff there exists $t \in T$ where $\ell(t) = a$ and there is $b \in B(t)$ s.t.

$$W((p,t),b) \subseteq M(p) \text{ and } W_I(p,t) > |M(p)| \text{ for all } p \in P, \text{ and}$$
$$M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b) \text{ for all } p \in P.$$

We denote the firing of a transition $t \in T$ in marking M reaching M' as $M \xrightarrow{t} M'$. Let $\to = \bigcup_{t \in T} \xrightarrow{t}$ and let $\to^*$ be the reflexive and transitive closure of $\to$.

Remark 1. To reason about model checking of CPNs, we need to have a finite representation of colored nets that can be passed as an input to an algorithm. One way to enforce such a representation is to assume that all color domains are finite and the semantic functions C, G, W and W_I are effectively computable.

Finally, let us define the notion of postset and preset of $p \in P$ as $p^\bullet = \{t \in T \mid \exists b \in \mathbb{B}.\ W((p,t),b) \neq \emptyset\}$ and ${}^\bullet p = \{t \in T \mid \exists b \in \mathbb{B}.\ W((t,p),b) \neq \emptyset\}$. Similarly, for a transition $t \in T$ we define $t^\bullet = \{p \in P \mid \exists b \in \mathbb{B}.\ W((t,p),b) \neq \emptyset\}$ and ${}^\bullet t = \{p \in P \mid \exists b \in \mathbb{B}.\ W((p,t),b) \neq \emptyset\}$. We also define the preset of inhibitor arcs as ${}^\circ t = \{p \in P \mid W_I(p,t) \neq \infty\}$.

2.2 P/T Nets

A Place/Transition (P/T) net is a CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ with one color $\mathbb{C} = \{\bullet\}$ and only one binding $\mathbb{B} = \{b_\epsilon\}$ s.t. every guard evaluates to true i.e. $G(t, b_\epsilon) = \mathit{true}$ for all $t \in T$ and every arc evaluates to a multiset over $\{\bullet\}$ i.e. $W((p,t), b_\epsilon) \in \mathcal{S}(\{\bullet\})$ and $W((t,p), b_\epsilon) \in \mathcal{S}(\{\bullet\})$ for all $p \in P$ and $t \in T$.

2.3 Integer Colored Petri Nets

An integer CPN (as used e.g. in the Model Checking Contest [18]) is a CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ where all colors are integer products i.e. $\mathbb{C} = \bigcup_{k \geq 1} (\mathbb{N}^0)^k$. We use interval ranges to describe sets of colors s.t. a tuple of ranges $([a_1, b_1], ..., [a_k, b_k])$ where $a_i, b_i \in \mathbb{N}^0$ for $i, 1 \leq i \leq k$, describes the set of colors $\{(c_1, ..., c_k) \mid a_i \leq c_i \leq b_i \text{ for all } 1 \leq i \leq k\}$. If the interval upperbound is smaller than the lowerbound, the interval range denotes the empty set and by $[a]$ we denote the singleton interval $[a, a]$. We use the set of variables $\mathcal{V} = \{x_1, ..., x_n\}$ to represent colors. Variables can be present on arcs and in guards. A binding $b : \mathcal{V} \to \mathbb{C}$ assigns colors to variables. We write $b \equiv \langle x_1 = c_1, ..., x_n = c_n \rangle$ for a binding where $b(x_i) = c_i$ for all i, $1 \leq i \leq n$. We now introduce the syntax of arc/guard expressions and its intuitive semantics by an example.

Figure 1 shows an integer CPN where places (circles) are associated with ranges. The initial marking contains five tokens (two of color 0 and three of color 2) in p_1 and two tokens of color 5 in place p_2. There is a guard on transition t (rectangle) that compares x with the integer 1 and restricts the valid bindings. We can see that the arc from t to p_3 creates a product of the integers x and y, where the value of x is decremented by one. We assume that all ranges are cyclic, meaning that the predecessor of 0 in the color set A is 2. Figure 1 also shows an example of transition firing. Markings are written as formal sums showing how many tokens of what colors are in the different places. The transition t can fire only once, as the inhibitor arc (for unlabelled inhibitor arcs we assume the default weight 1) from place p_3 to transition t inhibits the second transition firing.

The CPN model used in Model Checking Contest [18] further uses color types called dots and cyclic enumerations—these can be easily translated to integer ranges. All examples in this paper are expressed in integer CPN syntax.

2.4 Unfolding

CPNs with finite color domains can be *unfolded* into an equivalent P/T net [15]. Each place p is unfolded into $|C(p)|$ places, a transition is made for each legal

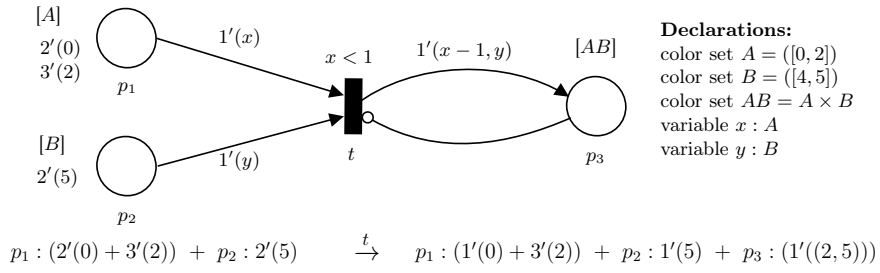


Fig. 1: Integer CPN and transition firing under the binding $\langle x = 0, y = 5 \rangle$

binding and we translate the multiset of colors on the arc to a multiset over $\bullet$. We now provide a formal definition of unfolding in our syntax, following the approach from [4, 17].

For each place connected to an inhibitor arc, we create a fresh summation place that contains the sum of tokens across the rest of the unfolded places. The summation places is created to ensure that inhibitor arcs functions correctly after unfolding.

Definition 2 (Unfolding). *Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN. The unfolded P/T net $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$ is given by*

1. $P^u = \{p(c) \mid p \in P \wedge c \in C(p)\} \cup \{p(\boldsymbol{sum}) \mid t \in T, p \in {}^\circ t\}$,
2. $T^u = \bigcup_{t \in T} \bigcup_{b \in B(t)} t(b)$,
3. $\mathbb{C}^u = \{\bullet\}$,
4. $\mathbb{B}^u = \{b_\epsilon\}$,
5. $C^u(p(c)) = \{\bullet\}$ *for all* $p(c) \in P^u$,
6. $G^u(t(b), b_\epsilon) =$ *true for all* $t(b) \in T^u$,
7. $W^u((p(c), t(b)), b) = W((p, t), b)(c)'(\bullet)$ *and* $W^u((t(b), p(c)), b) = W((t, p), b)(c)'(\bullet)$ *for all* $p(c) \in P^u$ *and* $t(b) \in T^u$, *and* $W^u((p(\boldsymbol{sum}), t(b)), b) = |W((p, t), b)|'(\bullet)$ *and* $W^u((t(b), p(\boldsymbol{sum})), b) = |W((t, p), b)|'(\bullet)$ *for all* $p(\boldsymbol{sum}) \in P^u$ *and* $t(b) \in T^u$,
8. $W_I^u(p(\boldsymbol{sum}), t(b)) = W_I(p, t)$ *for all* $p(\boldsymbol{sum}) \in P^u$ *and* $t(b) \in T^u$, *and*
9. $M_0^u(p(c)) = M_0(p)(c)'(\bullet)$ *for all* $p(c) \in P^u$ *and* $M_0^u(p(\boldsymbol{sum})) = |M_0(p)|'(\bullet)$ *for all* $p(\boldsymbol{sum}) \in P^u$

where $p(\boldsymbol{sum})$ denotes the sum of all tokens regardless of color for place p.

The theorem showing that the unfolded net is bisimilar to the original CPN was proved in [4, 17]; we only add a small optimization on the summation places.

Theorem 1 [4, 17]. *Given a CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ and the unfolded CPN $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$ then $M_0 \sim M_0^u$ with labeling function $\ell(t(b)) = t$ for all $t(b) \in T^u$.*

3 Color Quotienting

Unfolding a CPN without any further analysis will often lead to many unnecessary places and transitions. We shall now present our first technique that allows to group equivalently behaving colors into equivalence classes in order to reduce the number of colors and hence also to reduce the size of the unfolded net.

As an example consider the CPN in Figure 2a, the unfolded version of this net adds five places for both p_1 and p_2. However, we see that in p_1 all colors greater than or equal to 3 behave exactly the same throughout the net and can thus be represented by a single color. We can thus *quotient* the CPN by *partitioning* the color domain of each place into a number of *equivalence classes* of colors s.t. the colors behaving equivalently are represented by the same equivalence class. Using this approach we can construct a bisimilar CPN seen in Figure 2b where the color ($[3, 5]$) now represents all colors greater than or equal to 3.

Such a reduction in the number of colors is possible to include already during the design of a CPN model, however, the models may look less intuitive for human modeller or the nets can be auto-generated and hence contain redundant/equivalent colors as observed in the benchmark of CPN models from the annual Model Checking Contest benchmark [18].

We thus introduce *color partition* on places where all colors with similar behaviour in a given place are grouped into an *equivalence class*, denoted by θ. Let us assume a fixed CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$. A partition δ is a function $\delta : P \to 2^{2^{\mathbb{C}}} \setminus \emptyset$ that for a place p returns the equivalence classes of $C(p)$ s.t. $(\bigcup_{\theta \in \delta(p)} \theta) = C(p)$ and $\theta_1 \cap \theta_2 = \emptyset$ for all $\theta_1, \theta_2 \in \delta(p)$ where $\theta_1 \neq \theta_2$.

Definition 3. *Given a partition δ and markings M and M', we write $M(p) \overset{\delta}{\equiv} M'(p)$ for a $p \in P$ iff for all $\theta \in \delta(p)$ it holds that $\sum_{c \in \theta} M(p)(c) = \sum_{c \in \theta} M'(p)(c)$. We write $M \overset{\delta}{\equiv} M'$ iff $M(p) \overset{\delta}{\equiv} M'(p)$ for all $p \in P$. A partition δ is* stable *if the relation $\overset{\delta}{\equiv}$ on markings induced by δ is a bisimulation.*

Consider the CPN in Figure 2a. The partition shown in the Figure 2c is not stable as demonstrated by the transition firing from M_1 and M_2 to M_1' and M_2' where $M_1 \overset{\delta}{\equiv} M_2$ but $M_1' \overset{\delta}{\not\equiv} M_2'$. Figure 2d shows an example of a stable partition (here we describe the partition with ranges in the same manner as in integer CPNs).

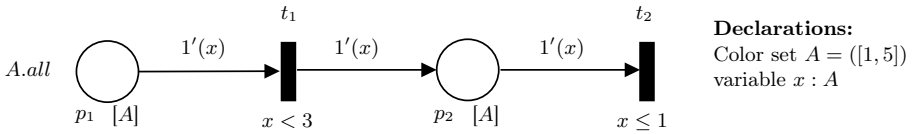
We now describe how a CPN may be quotiented using a stable partition. First, we define the notion of binding equivalence under a partition.

Definition 4. *Given a partition δ, a transition $t \in T$ and bindings $b, b' \in B(t)$, we write $b \overset{\delta,t}{\equiv} b'$ iff for all $p \in {}^\bullet t$ and for all $\theta \in \delta(p)$ it holds that*

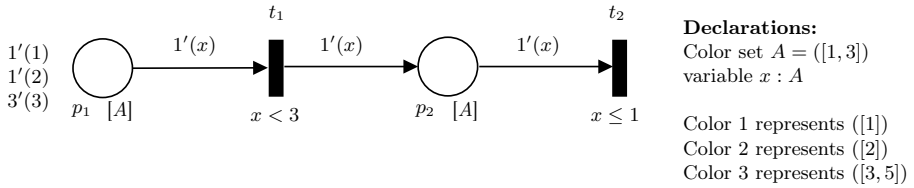
$$\sum_{c \in \theta} W((p,t),b)(c) = \sum_{c \in \theta} W((p,t),b')(c)$$

and for all $p \in t^\bullet$ and for all $\theta \in \delta(p)$ it holds that

$$\sum_{c \in \theta} W((t,p),b)(c) = \sum_{c \in \theta} W((t,p),b')(c).$$



(a) Example CPN



(b) Quotiented net from Figure 2a

$$\delta(p_1) = \{([1, 2]), ([3, 5])\}, \delta(p_2) = \{([1]), ([2, 5])\}$$

$$M_1 = p_1 : 1'(1) \xrightarrow{t_1} M_1' = p_2 : 1'(1)$$

$$M_1 \overset{\delta}{\equiv} M_2 \qquad\qquad M_1' \overset{\delta}{\not\equiv} M_2'$$

$$M_2 = p_1 : 1'(2) \xrightarrow{t_1} M_2' = p_2 : 1'(2)$$

(c) Example of an unstable partition δ and markings showing why it is unstable

$$\delta'(p_1) = \{([1]), ([2]), ([3, 5])\}, \delta'(p_2) = \{([1]), ([2, 5])\}$$

(d) Example of stable partition δ'

Fig. 2: Quotienting example

We can now define classes of equivalent bindings given a partition δ which are bindings that have the same behaviour for a given transition, formally $B^\delta(t) \overset{\text{def}}{=} \{[b]_t \mid b \in B(t)\}$ where $[b]_t = \{b' \mid b' \overset{\delta,t}{\equiv} b\}$.

For a given stable partition, we now construct a quotiented CPN where the set of colors are the equivalence classes of the stable partition and the set of bindings are the equivalence classes of bindings. As such, we rewrite the arc and guard evaluation functions to instead consider an equivalence class of bindings, which is possible since each binding in the equivalence class behaves equivalently.

Definition 5. *Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN and δ a stable partition of $\mathcal{N}$. The quotiented CPN $\mathcal{N}^\delta = (P, T, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ is defined by*

1. $\mathbb{C}^\delta = \bigcup_{p \in P} \delta(p)$
2. $\mathbb{B}^\delta = \biguplus_{t \in T} B^\delta(t)$.
3. $G^\delta(t, [b]_t) = G(t, b)$ *for all* $t \in T$ *and* $[b]_t \in B(t)$,
4. $C^\delta(p) = \delta(p)$ *for all* $p \in P$,

5. $W^\delta((p,t),[b]_t) = S$ *where* $S(\theta) = \sum_{c\in\theta} W((p,t),b)(c)$ *for all* $\theta \in \delta(p)$ *and* $W^\delta((t,p),[b]_t) = S$ *where* $S(\theta) = \sum_{c\in\theta} W((t,p),b)(c)$ *for all* $\theta \in \delta(p)$ *for all* $p \in P$, $t \in T$ *and* $[b]_t \in \mathbb{B}^\delta$,
6. $W_I^\delta(p,t) = W_I(p,t)$ *for all* $p \in P$ *and* $t \in T$, *and*
7. $M_0^\delta(p) = S$ *where* $S(\theta) = \sum_{c\in\theta} M_0(p)(c)$ *for all* $p \in P$ *and* $\theta \in \delta(p)$.

We can now present our main correctness theorem, stating that the original and quotiented colored nets are bisimilar.

Theorem 2. *Let* $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ *be a CPN,* δ *a stable partition and* $\mathcal{N}^\delta = (P^\delta, T^\delta, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ *the quotiented CPN. Then* $M_0 \sim M_0^\delta$.

3.1 Computing Stable Partitions

Our main challenge is how to efficiently compute a stable partition in order to apply the quotienting technique. To do so, we first define a partition refinement.

Definition 6. *Given two partitions* δ *and* δ' *we write* $\delta \geq \delta'$ *iff for all* $p \in P$ *and all* $\theta' \in \delta'(p)$ *there exists* $\theta \in \delta(p)$ *s.t.* $\theta' \subseteq \theta$. *Additionally, we write* $\delta > \delta'$ *if* $\delta \geq \delta'$ *and* $\delta' \neq \delta$.

Note that for any finite CPN as assumed in Remark 1, the refinement relation $>$ is well-founded as for any $\delta > \delta'$ the partition δ' has strictly more equivalence classes for at least one place $p \in P$. We now define also the union of two partitions as the smallest partition that has both of the partitions as refinements.

Definition 7. *Given two partitions* δ_1, δ_2 *and* $p \in P$, *let* $\leftrightarrow$ *be a relation over* $\delta_1(p) \cup \delta_2(p)$ *s.t.* $\theta \leftrightarrow \theta'$ *iff* $\theta \cap \theta' \neq \emptyset$ *where* $\theta, \theta' \in \delta_1(p) \cup \delta_2(p)$. *Let* $\leftrightarrow^*$ *be the reflexive, transitive closure of* $\leftrightarrow$ *and let* $[\theta] \stackrel{\text{def}}{=} \bigcup_{\theta'\in\delta_1(p)\cup\delta_2(p), \theta\leftrightarrow^*\theta'} \theta'$ *where* $\theta \in \delta_1(p) \cup \delta_2(p)$. *Finally, we define the partition union operator* $\sqcup$ *by* $(\delta_1 \sqcup \delta_2)(p) = \bigcup_{\theta\in\delta_1(p)\cup\delta_2(p)} \{[\theta]\}$ *for all* $p \in P$.

For example, assume some place p s.t. $C(p) = \{([1,5])\}$ and partitions δ_1 and δ_2 s.t. $\delta_1(p) = \{([1,2]),([3,4]),([5])\}$ and $\delta_2(p) = \{([1]),([2,3]),([4]),([5])\}$ then $(\delta_1 \sqcup \delta_2)(p) = \{([1,4]),([5])\}$.

Lemma 1. *Let* δ_1 *and* δ_2 *be two partitions. Then (i)* $\delta_1 \sqcup \delta_2 \geq \delta_1$ *and* $\delta_1 \sqcup \delta_2 \geq \delta_2$, *and (ii) if* δ_1 *and* δ_2 *are stable partitions then so is* $\delta_1 \sqcup \delta_2$.

The lemma above implies the existence of a unique maximum stable partition.

Theorem 3. *There is a unique maximum stable partition* δ *s.t.* $\delta \geq \delta'$ *for all stable partitions* δ'.

In order to provide an algorithm for computing a stable partition, we define the maximum arc size for a given CPN $\mathcal{N}$ as the function $max(\mathcal{N}) = \max_{p\in P, t\in T, b\in\mathbb{B}}(|W((p,t),b)|, |W((t,p),b)|)$. The set of all markings smaller than

Algorithm 1: *Stabilize*($\mathcal{N}$)

1 **Input**: $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$
2 **Output**: Stable partition δ
3 let $\delta(p) := \{C(p)\}$ for all $p \in P$
4 **for** $t \in T$ **do**
5 **while** $\exists M_1, M_2 \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{\delta}{\equiv} M_2 \wedge M_1 \overset{t}{\not\rightarrow} \wedge M_2 \overset{t}{\rightarrow}$ **do**
6 pick $\delta' < \delta$ s.t. $M_1 \overset{\delta'}{\not\equiv} M_2$; $\delta := \delta'$
7 **end**
8 **end**
9 let $\mathcal{Q} := P$ //Waiting list of places
10 **while** $\mathcal{Q} \neq \emptyset$ **do**
11 let $p \in \mathcal{Q}$; $\mathcal{Q} := \mathcal{Q} \setminus \{p\}$
12 **for** $t \in {}^\bullet p$ **do**
13 **if** $\exists M_1, M_2 \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{\delta}{\equiv} M_2.\exists M_1' \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{t}{\rightarrow}$
 $M_1' \wedge \forall M_2' \in \mathbb{M}^{bounded}(\mathcal{N}).M_2 \overset{t}{\rightarrow} M_2' \wedge M_1'(p) \overset{\delta}{\not\equiv} M_2'(p)$ **then**
14 pick $\delta' < \delta$ s.t. $M_1 \overset{\delta'}{\not\equiv} M_2$ and $\delta'(p') = \delta(p')$ for all $p' \in P \setminus {}^\bullet t$
15 $\mathcal{Q} := \mathcal{Q} \cup \{p' \mid \delta'(p') \neq \delta(p')\}$; $\delta := \delta'$
16 **end**
17 **end**
18 **end**
19 **return** δ

the *max* arc size over $\mathcal{N}$ is defined by $\mathbb{M}^{bounded}(\mathcal{N}) = \{M \in \mathbb{M}(\mathcal{N}) \mid |M(p)| \leq max(\mathcal{N})$ for all $p \in P\}$. As such, $\mathbb{M}^{bounded}(\mathcal{N})$ is a finite set of all bounded markings of $\mathcal{N}$ with cardinality less than or equal to $max(\mathcal{N})$.

Algorithm 1 now gives a procedure for computing a stable partition over a given CPN. It starts with an initial partition where every color in the color domain is in the same equivalence class for each place. The algorithm is then split into two parts. The first part from line 4 to 8 creates an initial partition applying the guard restrictions to the input places of the transitions. The second part from line 10 to 18 back propagates the guard restrictions throughout the net s.t. only colors that behave the same are quotiented together. Depending on the choices in lines 6 and 14, the algorithm may return the maximum stable partition, however in the practical implementation this is not guaranteed due to an approximation of the guard/arc expression analysis.

Theorem 4. *Given a CPN $\mathcal{N}$, the algorithm Stabilize($\mathcal{N}$) terminates and returns a stable partition of $\mathcal{N}$.*

3.2 Stable Partition Algorithm for Integer CPNs

The *Stabilize* computation presented in Algorithm 1 can be used to find a stable partition for any finite CPN. However, implementation-wise it is inefficient to represent every color in a given equivalence class individually. Hence, for integer

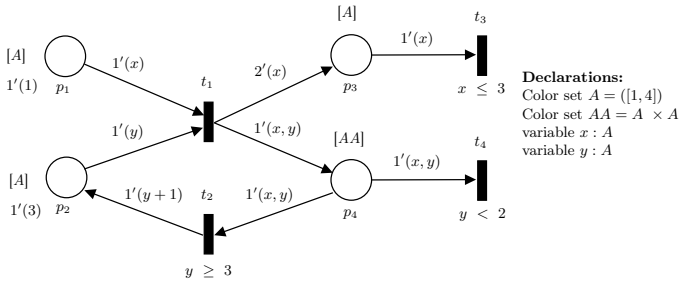


Fig. 3: Example CPN

Iteration	p_1	p_2	p_3	p_4	$\mathcal{Q}$
0	$\{([1,4])\}$	$\{([1,4])\}$	$\{([1,3]),([4])\}$	$\{([1,4],[1]),([1,4],[2]),([1,4],[3,4])\}$	$\{p_1,p_2,p_3,p_4\}$
1, $p=p_3$	$\{([1,3]),([4])\}$	-	-	-	$\{p_1,p_2,p_4\}$
2, $p=p_4$	-	$\{([1]),([2]),([3,4])\}$	-	-	$\{p_1,p_2\}$
3, $p=p_2$	-	-	-	$\{([1,4],[1]),([1,4],[2]),$ $([1,4],[3]),([1,4],[4])\}$	$\{p_1,p_4\}$
4, $p=p_4$	-	$\{([1]),([2]),([3]),([4])\}$	-	-	$\{p_1,p_2\}$
5, $p=p_2$	-	-	-	-	$\{p_1\}$
6, $p=p_1$	-	-	-	-	$\{\}$

Table 1: Stages of δ throughout Algorithm 1 for CPN in Figure 3. The 0th iteration is the state of δ just before the while loop begins. The symbol '-' indicates that the value is the same as in the previous row.

Iteration	p_1	p_2	p_3	p_4
0, $\alpha=\alpha_0$	$\{([1])\}$	$\{([3])\}$	$\{\}$	$\{\}$
1, $t=t_1$	-	-	$\{([1])\}$	$\{([1],[3])\}$
2, $t=t_2$	-	$\{([3,4])\}$	-	-
3, $t=t_1$	-	-	-	$\{([1],[3,4])\}$
4, $t=t_2$	-	$\{([3,4]),([1])\}$	-	-
5, $t=t_1$	-	-	-	$\{([1],[3,4]),([1],[1])\}$
6, $t=t_2$	-	$\{([1,4])\}$	-	-
7, $t=t_1$	-	-	-	$\{([1],[1,4])\}$

Table 2: Stages of α when computing the fixed point of E for the CPN in Figure 3. The symbol '-' indicates that the value is the same as in the previous row.

CPN we represent an equivalence class as a tuple of ranges. As an example of computing stable partitions with Algorithm 1, consider the integer CPN in Figure 3. Table 1 shows the different stages that δ undergoes in order to become stable. In iteration 0, the guard restrictions from the first for-loop are applied, followed by the iterations of the main while-loop. In our implementation, we do not iterate through every bounded marking and we instead (for efficiency reasons) statically analyze the places, arcs and guards in order to partition the color sets. For example, in iteration number 1, we consider the place p_3 and we can see that the colors in the range $[1,3]$ must be distinguished from the color 4. This partitioning propagates back to the place p_1 as firing the transition t_1 moves tokens from p_1 to p_3 without changing its color.

4 Color Approximation

We now introduce another technique for safely overapproximating what colors can be present in each place of a CPN. Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a fixed CPN for the rest of this section. A *color approximation* is a function $\alpha : P \to 2^{\mathbb{C}}$ where $\alpha(p)$ approximates the possible colors in place $p \in P$ s.t. $\alpha(p) \subseteq C(p)$. Let $\mathbb{A}$ be the set of all color approximations. For a marking M and color approximation α, we write $M \subseteq \alpha$ iff $\mathit{set}(M(p)) \subseteq \alpha(p)$ for all $p \in P$. A *color expansion* is a function $E : \mathbb{A} \to \mathbb{A}$ defined by

$$E(\alpha)(p) = \begin{cases} \alpha(p) \cup \mathit{set}(W((t,p),b)) & \text{if } \exists t \in T. \exists b \in B(t). \\ & \mathit{set}(W((p,t),b)) \subseteq \alpha(p) \\ \alpha(p) & \text{otherwise.} \end{cases}$$

A color expansion iteratively expands the possible colors that exist in each place and obviously preserves the following property.

Lemma 2. *Let α be a color approximation then $\alpha(p) \subseteq E(\alpha)(p)$ for all $p \in P$.*

Let α_0 be the initial approximation such that $\alpha_0(p) \stackrel{\text{def}}{=} \mathit{set}(M_0(p))$ for all $p \in P$. Since E is a monotonic function on a complete lattice, we can compute its minimum fixed point and formulate the following key theorem.

Theorem 5. *Let α be a minimum fixed point of E such that $\alpha_0(p) \subseteq \alpha(p)$ for all $p \in P$. If $M_0 \to^* M$ then $M \subseteq \alpha$.*

Given a color approximation α satisfying the preconditions of Theorem 5, we can now construct a reduced CPN $\mathcal{N}^\alpha = (P, T, \mathbb{C}, \mathbb{B}, C^\alpha, G, W, W_I, M_0)$ where $C^\alpha(p) = \alpha(p)$ for all $p \in P$. The net $\mathcal{N}^\alpha$ can hence have possibly smaller set of colors in its color domains and it satisfies the following theorem.

Theorem 6. *The reachable fragments from the initial marking M_0 of the LTSs generated by $\mathcal{N}$ and $\mathcal{N}^\alpha$ are isomorphic.*

4.1 Computing Color Approximation on Integer CPNs

As with color quotienting, representing each color individually becomes inefficient. We thus employ integer ranges to represent color approximations. Consider the approximation α where $\alpha(p) = \{(1,2), (2,2), (3,2), (5,6), (5,7)\}$ are possible colors (pairs of integers) in the place p; this can be more compactly represented as a set of tuples of ranges $\{([1,3],[2,2]), ([5,5],[6,7])\}$.

However, computing the minimum fixed point of E using ranges is not as trivial as using complete color sets. To do so, we need to compute new ranges depending on arcs and guards. We demonstrate this on the CPN in Figure 3. Table 2 shows the computation of the minimum fixed point of E, starting from the initial approximation α_0. For example, in iteration number 5, we check if firing transition t_1 can produce any additional tokens to the places p_3 and p_4. Clearly, there is no change to the possible token colors in p_3 as $\alpha(p_1)$ did not change, however the addition of the integer range [1] to $\alpha(p_2)$ in the previous iteration now allows us to produce a new token color $(1,1)$ into p_4 and hence we add the singleton range $([1],[1])$ to $\alpha(p_4)$.

	Spike	Tapaal	A	ITS	B	MCC	A+B	Total
Unfolded nets	172	174	199	202	204	205	207	208

Table 3: Number of unfolded nets for each unfolder

5 Experiments

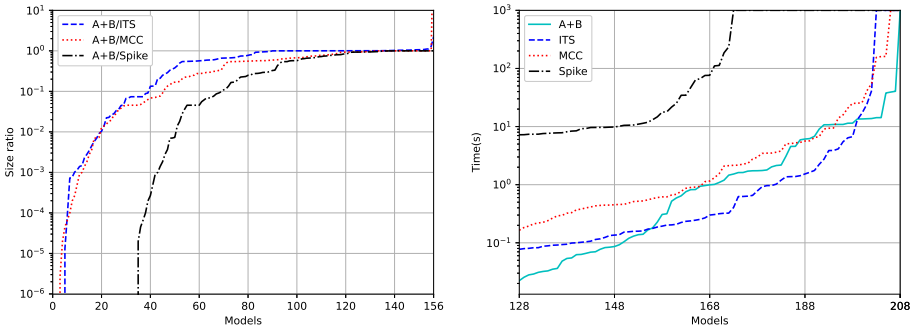
We implemented the quotienting method from Section 3 as well as the color approximation method from Section 4 in C++ as an extension to the verification engine *verifypn* [13] from the TAPAAL toolchain [7]. We also implemented the method of variable symmetry identification inspired by its use in ITS-Tools [25]; the effect of this method is marginal as it additionally reduces the size of the unfolded net only on a few instances.

We perform experiments comparing several different approaches; the quotienting approach (method A), the color approximation (method B) and the combination of quotienting, symmetric variables and color approximation (method A+B) against the unfolder MCC [6] (used also by TINA [1] and LoLA [27]), ITSTools unfolder [24] and Spike unfolder [3] (also used by MARCIE [10] and Snoopy [9]) and the previous *verifypn* TAPAAL unfolder (revision 226) referred to as Tapaal. We compare the tools on the complete set of CPN nets and queries from 2020 Model Checking Contest [18]. The experiments are conducted on a compute cluster running Linux version 5.8.0-2, where each experiment is conducted on a AMD Epyc 7551 processor with a 15 GB memory limit and 5 minute timeout. A repeatability package is available in [2].

Table 3 shows for each of the unfolders the number of unfolded nets within the memory/time limit. The last column shows the total number of unfolded nets by all tools combined. The single net that we cannot unfold is FamilyReunion3000 which was unfolded by MCC, though we can unfold it given 3 extra minutes. Our method A+B can unfold 3 nets that no other tool can unfold; DrinkVendingMachine48, 72, 96. This is directly attributed to method A.

The comparison of the sizes (total number of transitions and places) of unfolded nets is done by plotting the ratios between the size produced by our A+B method and the competing unfolder. Figure 4a shows the size ratios where at least one comparison is not equal to 1. We see that our method has a significantly smaller size ratio for 88 colored nets, sometimes reducing the nets by several orders of magnitude. In a few cases ITS-Tools is able to unfold to a smaller net than our method due to their post-reductions. This is most prevalent on VehicularWifi which they unfold to the size of 38429 objects while we create a net of size 85835. The figure also shows one net unfolded by MCC where we timed out.

As our method outperforms the state-of-the-art unfolders w.r.t. the size of the unfolded nets, the question is whether the overhead of the advanced static analysis does not kill the benefits. Fortunately, this is not the case as shown in Figure 4b where the 80 slowest running times (independently sorted in non-decreasing order) for each tool are depicted. The plots show that ITS, MCC and our unfolder are close in performance, while Spike is slower. ITS-Tools is



(a) Size-ratios (nondecreasingly ordered) (b) Worst 80 unfolding times

Fig. 4: Unfolding size and unfolding time comparison

generally fast on the nets that are unfolded in less than 10 seconds, however it becomes gradually slower and has problems unfolding the larger nets. The MCC unfolder and our method are similar in performance, except for the largest instances where we are faster.

The overall conclusion is that our advanced analyses adds only a little overhead while significantly decreasing the size of the unfolded nets. This is also confirmed by the number of answered reachability, CTL and LTL queries from the 2020 Model Checking Contest benchmark. The colored nets and queries are unfolded by the different tools and then verified by the TAPAAL engine. Here our unfolding method allows TAPAAL to answer in total 81.7% of all queries whereas the MCC unfolder can answer 76.9% and ITS-Tools unfolder 76.1% of all queries.

6 Conclusion

We presented two complementary methods for reducing the unfolding size of colored Petri nets (CPN). Both methods are proved correct and implemented in an open-source verification engine of the tool TAPAAL. Experimental results show a significant improvement in the size of unfolded nets, compared to state-of-the-art tools, without compromising the unfolding speed. The actual verification on the models and queries from the 2020 Model Checking Contest shows that our unfolding technique allows us to solve 4.8% more queries compared to the second best competing tool. In future work, we plan to combine our approach with structural reduction techniques.

Acknowledgments. We would like to thank Yann Thierry-Mieg for his answers and modifications to the ITS-Tools, Silvano Dal Zilio for his answers/additions concerning the MCC unfolder and Monika Heiner and Christian Rohr for their answers concerning the tools Snoopie, Marcie and Spike.

Bibliography

[1] B. Berthomieu, P.-O Ribet, and F. Vernadat. The tool TINA — construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42:2741–2756, 2004. doi.org/10.1080/00207540412331312688.

[2] A. Bilgram, P.G. Jensen, T. Pedersen, J. Srba, and P.H. Taankvist. Repeatability Package for: Improvements in Unfolding of Colored Petri Nets, 2021. https://doi.org/10.5281/zenodo.5255603.

[3] J. Chodak and M. Heiner. Spike - Reproducible Simulation Experiments with Configuration File Branching. In *Computational Methods in Systems Biology*, pages 315–321, Cham, 2019. Springer International Publishing. http://doi.org/10.1007%2F978-3-030-31304-3_19.

[4] N. Christensen, M. Glavind, S. Schmid, and J. Srba. Latte: Improving the Latency of Transiently Consistent Network Update Schedules. *SIGMETRICS Perform. Eval. Rev.*, 48(3):14–26, 2021. doi.org/10.1145/3453953.3453957.

[5] A. Ciaghi, K. Weldemariam, A. Villafiorita, and F. Kessler. Law Modeling with Ontological Support and BPMN: a Case Study. *In CYBERLAWS 2011, The Second International Conference on Technical and Legal Aspects of the e-Society*, pages 29–34, 2011.

[6] S. Dal Zilio. MCC: A Tool for Unfolding Colored Petri Nets in PNML Format. In *Application and Theory of Petri Nets and Concurrency*, pages 426–435, Cham, 2020. Springer International Publishing. http://doi.org/10.1007/978-3-030-51831-8_23.

[7] A. David, L. Jacobsen, M. Jacobsen, K.Y. Jørgensen, M.H. Møller, and J. Srba. 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)*, volume 7214 of *LNCS*, pages 492–497. Springer-Verlag, 2012. doi.org/10.1007/978-3-642-28756-5_36.

[8] E. Gilberto Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis. 30 Years of GreatSPN. In *Principles of Performance and Reliability Modeling and Evaluation*, volume 7927, pages 227–254. Springer, 2016. doi.org/10.1007/978-3-319-30599-8_9.

[9] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick. Snoopy – A Unifying Petri Net Tool. In *Application and Theory of Petri Nets*, pages 398–407, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi.org/10.1007/978-3-642-31131-4_22.

[10] M. Heiner, C. Rohr, and M. Schwarick. MARCIE - Model Checking and Reachability Analysis Done Efficiently. In *Application and Theory of Petri Nets and Concurrency*, pages 389–399, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-38697-8_21.

[11] L.M. Hillah. A hot drink vending machine. https://mcc.lip6.fr/pdf/DrinkVendingMachine-form.pdf, 2021.

[12] L.M. Hillah. Family Reunion. https://mcc.lip6.fr/pdf/FamilyReunion-form.pdf, 2021.

[13] J.F. Jensen, T. Nielsen, L.K. Oestergaard, and J. Srba. TAPAAL and reachability analysis of P/T nets. *LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC)*, 9930:307–318, 2016.

[14] K. Jensen. Coloured Petri Nets and the Invariant-Method. *Theoretical Computer Science*, 14:317–336, 1981. doi.org/10.1016/0304-3975(81)90049-9.

[15] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Volume 1*, volume 1. Springer-Verlag Berlin Heidelberg, 2 edition, 1996. https://www.springer.com/gp/book/9783540609438.

[16] K. Jensen and L. M. Kristensen. *Coloured Petri Nets, Modelling and Validation of Concurrent Systems*. Springer-Verlag Berlin Heidelberg, 1 edition, 2009. doi.org/10.1007/b95112.

[17] A.H. Klostergaard. Efficient Unfolding and Approximation of Colored Petri Nets with Inhibitor Arcs. Master's thesis, Department of Computer Science, Aalborg University, 2018. https://projekter.aau.dk/projekter/files/281079031/main.pdf.

[18] F. Kordon, H. Garavel, L. M. Hillah, F. Hulin-Hubard, E. Amparore, B. Berthomieu, S. Biswal, D. Donatelli, F. Galla, G. Ciardo, S. Dal Zilio, P. G. Jensen, C. He, D. Le Botlan, S. Li, A. Miner, J. Srba, and Y. Thierry-Mieg. Complete Results for the 2020 Edition of the Model Checking Contest. http://mcc.lip6.fr/2020/results.php, 2020.

[19] F. Liu, M. Heiner, and M. Yang. An efficient method for unfolding colored Petri nets. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12, 2012. doi.org/10.1109/WSC.2012.6465203.

[20] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989. doi.org/10.1109/5.24143.

[21] R. Muschevici, J. Proença, and D. Clarke. Modular Modelling of Software Product Lines with Feature Nets. In *Software Engineering and Formal Methods*, pages 318–333, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-24690-6_22.

[22] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[23] M. Schwarick, C. Rohr, F. Liu, G. Assaf, J. Chodak, and M. Heiner. *Efficient Unfolding of Coloured Petri Nets Using Interval Decision Diagrams*, pages 324–344. Application and Theory of Petri Nets and Concurrency, 2020. http://doi.org/10.1007/978-3-030-51831-8_16.

[24] Y. Thierry-Mieg. Symbolic Model-Checking Using ITS-Tools. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 231–237, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. http://doi.org/10.1007%2F978-3-662-46681-0_20.

[25] Y. Thierry-Mieg. Personal correspondence with Y. Thierry-Mieg, 2021.

[26] Y. Thierry-Mieg, C. Dutheillet, and I. Mounier. Automatic symmetry detection in well-formed nets. In W. van der Aalst and Eike Best, editors, *Applications and Theory of Petri Nets 2003*, pages 82–101, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[27] K. Wolf. Petri Net Model Checking with LoLA 2. In *Application and Theory of Petri Nets and Concurrency*, pages 351–362, Cham, 2018. Springer International Publishing. doi.org/10.1007/978-3-319-91268-4_18.

A Proofs for Section 2 (Preliminaries)

A.1 Integer Colored Petri Nets

An integer CPN is a CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ where all colors are integer products i.e. $\mathbb{C} = \bigcup_{k \geq 1} (\mathbb{N}^0)^k$.

We introduce the notion of ranges to describe the place color type s.t. a tuple of ranges $([a_1, b_1], ..., [a_k, b_k])$ where $a_i, b_i \in \mathbb{N}^0$ for $i, 1 \leq i \leq k$ describes a set of colors given by the following semantics:

$$[\![([a_1, b_1], ..., [a_k, b_k])]\!] = \{(c_1, ..., c_k) \mid a_i \leq c_i \leq b_i \textit{ for all } 1 \leq i \leq k\}$$

As an example, consider the place color type of some place p as $C(p) = [\![([1, 2], [6, 7])]\!]$ describing the set of colors $\{(1, 6), (1, 7), (2, 6), (2, 7)\}$. For simplicity, we omit the semantic meaning and simply denote $([1, 2], [6, 7])$ as the colors of $[\![([1, 2], [6, 7])]\!]$. We also represent singleton intervals as only one number, e.g. $[2, 2]$ is represented as $[2]$. Lastly, note that for $a, b \in \mathbb{N}^0$ where $a > b$ then $([a, b]) = \emptyset$.

In integer CPNs variables are used to represent colors. They can be present on arcs and in guards. We denote the set of all variables as $\mathcal{V}$. Bindings in integer CPNs denote a concrete value assignment of variables s.t. a binding is defined as a function $b : \mathcal{V} \to \mathbb{C}$ giving the value of a variable under a binding. We denote a concrete binding of variables $\{x_1, ..., x_n\}$ as $b = \langle x_1 = c_1, ..., x_n = c_n \rangle$ where $b(x_i) = c_i$ for all i, $1 \leq i \leq n$.

In integer CPNs each arc $(P \times T) \cup (T \times P)$ excluding inhibitor arcs is assigned an arc expression a given by the syntax:

$$\tau ::= c \mid x \mid x \pm s$$
$$a ::= n'(\tau_1, ..., \tau_k) \mid a_1 \pm a_2 \mid n \cdot a$$

where $c \in \mathbb{C}$, $x \in \mathcal{V}$, $s \in \mathbb{N}^{>0}$, $n \in \mathbb{N}^{>0}$ and $\pm ::= + \mid -$.

The semantics of arc expressions are straightforward and are demonstrated by an example.

Example 1. Let $a = 1'(x - 1) + 1'(y + 1) + 1'(z)$ be an arc expression and $b_1 = \langle x = 3, y = 3, z = 1 \rangle$ and $b_2 = \langle x = 1, y = 2, z = 2 \rangle$ be bindings with range $([1, 3])$ over variables x, y and z. The semantics is defined as multisets where $W(a, b_1) = 1'(2) + 2'(1)$ since the colors are cyclic in nature s.t. $3 + 1 = 1$ and $W(a, b_2) = 2'(3) + 1'(2)$ because $1 - 1 = 3$.

Guards in integer CPNs are expressed by the following syntax:

$$\gamma ::= \textit{true} \mid \textit{false} \mid \neg\gamma \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1 \vee \gamma_2 \mid \tau_1 \bowtie \tau_2$$

where $\bowtie ::= < \mid \leq \mid > \mid \geq \mid = \mid \neq$. The semantics of guards are also straightforward and they evaluate to truth values. They are again demonstrated by an example.

Example 2. Let $g = x > 2 \wedge y = 2 \vee z + 2 = 3$ be a guard and $b_1 = \langle x = 3, y = 3, z = 1 \rangle$ and $b_2 = \langle x = 1, y = 2, z = 2 \rangle$ be bindings with range $([1, 3])$ over variables x, y and z. The semantics is defined as $G(g, b_1) = \mathit{true}$ and $G(g, b_2) = \mathit{false}$.

Figure 5a shows an example of an integer CPN. We see all places are associated with a set of ranges, i.e. $C(p_1) = A$ noted $[A]$. We also see that there is a guard on transition t that compares x with the integer 1. Lastly, we see that the post arc from t to p_3 creates a product of the integers x and y, where the value of x is decremented by one. This means that the value of $x - 1$ is the previous color in color set A. Note that the previous color for 0 is 2 as the color sets are cyclic in nature as mentioned above. Figure 5b shows an example of transition firing in Figure 5a. The transition may only fire once, as the inhibitor arc from place p_3 to transition t inhibits the transition when there is at least 1 token in p_3.

The Model Checking Contest [18] further includes color types called dots and cyclic enumerations which are excluded from these definitions, as these can be trivially translated to tuples of integer ranges. For dots, $\{\bullet\}$, it is simply translated to the color domain $([1])$ and for a cyclic enumeration with elements $\{e_1, e_2, ..., e_n\}$ it is translated to the integer colors corresponding to the indices of the cyclic enumeration, $([1, n])$. Furthermore, the Model Checking Contest uses *.all* expressions, which creates one of each color in a color domain. This can be translated to the multiset with one of each color. As an example, consider the color set $A = ([0, 2])$ from Figure 5a then $A.all = 1'(1) + 1'(2) + 1'(3)$.

Consider the CPN in Figure 5a. The unfolded version of this can be seen in Figure 5c. We see that each place of the CPN is unfolded to a new place for every color in the color type of the place as well as a ***sum*** place for p_3. Additionally, the transition is unfolded to a new transition for each legal binding.

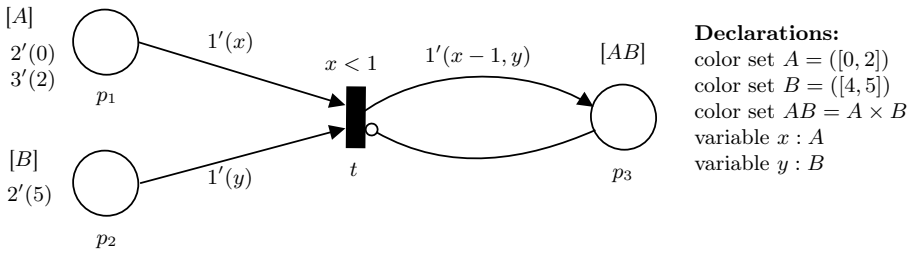
B Proofs for Section 3 (Color Quotienting)

Theorem 2. *Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN, δ a stable partition and $\mathcal{N}^\delta = (P^\delta, T^\delta, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ the quotiented CPN. Then $M_0 \sim M_0^\delta$.*

Proof. Let $R = \{(M, M^\delta) \mid \sum_{c \in \theta} M(p)(c) = M^\delta(p)(\theta)$ for all $p \in P$ and all $\theta \in \delta(p)\}$.

We first notice that $(M_0, M_0^\delta) \in R$ by Item 7 in Definition 5. We then show that R is a bisimulation.

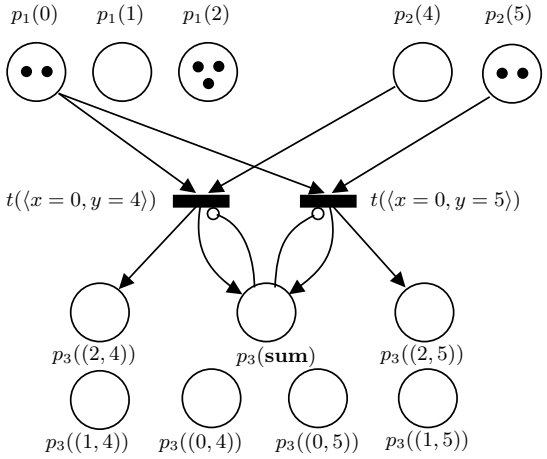
Assume $(M, M^\delta) \in R$ and $t \in T$ s.t. $M \xrightarrow{t} M'$ under binding $b \in B(t)$, we want to show that $M^\delta \xrightarrow{t} M^{\delta\prime}$ under binding $[b]_t \in B(t)$ s.t. $(M', M^{\delta\prime}) \in R$. As



(a) Example of Integer CPN
binding: $\langle x = 0, y = 5 \rangle$

$$p_1 : (2'(0) + 3'(2)) \ + \ p_2 : 2'(5) \quad \xrightarrow{t} \quad p_1 : (1'(0) + 3'(2)) \ + \ p_2 : 1'(5) \ + \ p_3 : (1'((2,5)))$$

(b) Transition firing in Figure 5a



(c) Unfolding of the CPN in Figure 5a to a P/T net

Fig. 5: Integer CPN and unfolding example

such, we need to prove the following:

- **(a)** $W^\delta((p,t),[b]_t) \subseteq M^\delta(p)$ for all $p \in P$
- **(b)** $W_I^\delta(p,t) > |M^\delta(p)|$ for all $p \in P$
- **(c)** $(M', M^{\delta\prime}) \in R$ *where* $M^{\delta\prime}(p) = (M^\delta(p) \setminus W^\delta((p,t),[b]_t)) \uplus W^\delta((t,p),[b]_t)$ for all $p \in P$

(a) We start by showing $W^\delta((p,t),[b]_t) \subseteq M^\delta(p)$ for all $p \in P$. Firstly, because $(M, M^\delta) \in R$ we know that

$$\textstyle\sum_{c\in\theta} M(p)(c) = M^\delta(p)(\theta) \text{ for all } p \in P \text{ and all } \theta \in \delta(p). \tag{1}$$

Since $W((p,t),b) \subseteq M(p)$ we know for all $c \in C(p)$ that $W((p,t),b))(c) \leq M(p)(c)$ by Multiset Definition of $\subseteq$ which implies that:

$$\textstyle\sum_{c\in\theta} W((p,t),b))(c) \leq \sum_{c\in\theta} M(p)(c) \tag{2}$$

for all $\theta \in \delta(p)$. We then want to show that $W^\delta((p,t),[b]_t) \subseteq M^\delta(p)$ i.e. $W^\delta((p,t),[b]_t)(\theta) \leq M^\delta(p)(\theta)$ for all $\theta \in \delta(p)$:

$W^\delta((p,t),[b]_t)(\theta)$	$=$	Substitute by Def. 5 Item 5
$\sum_{c\in\theta} W((p,t),b))(c)$	$\leq$	By Equation (2)
$\sum_{c\in\theta} M(p)(c)$	$=$	By Equation (1)
$M^\delta(p)(\theta)$		

for all $p \in P$, all $\theta \in \delta(p)$ and $b \in B(t)$.

(b) Next we show $W_I^\delta((p,t),[b]_t) > |M^\delta(p)|$. We know that

$$W_I(p,t) > |M(p)| \tag{3}$$

by definition of CPN semantics since $M \xrightarrow{t} M'$. We then show that:

$W_I^\delta(p,t)$	$=$	Substitute by Def. 5 Item 6
$W_I(p,t)$	$>$	Equation (3)
$\lvert M(p)\rvert$	$=$	Multiset Def.
$\sum_{c\in C(p)} M(p)(c)$	$=$	Since $(\bigcup_{\theta\in\delta(p)} \theta) = C(p)$
$\sum_{\theta\in\delta(p)} \sum_{c\in\theta} M(p)(c)$	$=$	By Equation (1)
$\sum_{\theta\in\delta(p)} M^\delta(p)(\theta)$	$=$	Multiset Def.
$\lvert M^\delta(p)\rvert$		

for all $p \in P, \theta \in \delta(p)$.

(c) Lastly, we show that $(M', M^{\delta\prime}) \in R$. Assume $p \in P$, $b \in B(t)$ and equivalence class $[b]_t$. We know that $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$ and $M^{\delta\prime}(p) = (M^\delta(p) \setminus W^\delta((p,t),[b]_t)) \uplus W^\delta((t,p),[b]_t)$ and we need to show that $\sum_{c\in\theta} M'(p)(c) = M^{\delta\prime}(p)(\theta)$ for all $\theta \in \delta(p)$:

$$
\begin{array}{ll}
\sum_{c\in\theta} M'(p)(c) & = \text{by def. of CPN semantics} \\
\sum_{c\in\theta}(M(p) \setminus W((p,t),b) \uplus W((t,p),b))(c) & = \text{Substitute by multiset definitions and by enabledness of } t \\
\sum_{c\in\theta} M(p)(c) - \sum_{c\in\theta} W((p,t),b)(c) + \sum_{c\in\theta} W((t,p),b)(c) & = \text{Substitute by Def. 5 Item 5} \\
\sum_{c\in\theta} M(p)(c) - W^\delta((p,t),[b]_t)(\theta) + W^\delta((t,p),[b]_t)(\theta) & = \text{Since } (M, M^\delta) \in R \\
M^\delta(p)(\theta) - W^\delta((p,t),[b]_t)(\theta) + W^\delta((t,p),[b]_t)(\theta) & = \text{by definition of CPN semantics} \\
M^{\delta\prime}(p)(\theta) &
\end{array}
$$

We then have to show that the same is the case for the opposite direction s.t. assume $(M, M^\delta) \in R$ and $t \in T$ s.t. $M^\delta \xrightarrow{t} M^{\delta\prime}$, we want to show that $M \xrightarrow{t} M'$, i.e. $W((p,t),b) \subseteq M(p)$ and $W_I(p,t) > M(p)$ for all $p \in P$ where $b \in B(t)$ and $(M', M^{\delta\prime}) \in R$. As such, we want to show that:

- **(d)** $W((p,t),b) \subseteq M(p)$ for all $p \in P$
- **(e)** $W_I(p,t) > |M(p)|$ for all $p \in P$
- **(f)** $(M', M^{\delta\prime}) \in R$ where $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$ for all $p \in P$

First notice that **(e)** and **(f)** can simply be showed by the same argumentation as **(b)** and **(c)**.

(d) We show that $W((p,t),b) \subseteq M(p)$ for all $p \in P$. From **(a)** we know that $W^\delta((p,t),[b]_t) \subseteq M^\delta(p)$ which implies $\sum_{c\in\theta} W((p,t),b))(c) \leq \sum_{c\in\theta} M(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$.

Hence observe that there is a marking M_1 s.t. $\sum_{c\in\theta} M_1(p)(c) = M^\delta(p)(\theta)$ and $\sum_{c\in\theta} W((p,t),b)(c) \leq \sum_{c\in\theta} M_1(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$. Clearly t is enabled in M_1 since $W((p,t),b) \subseteq M_1(p)$ for all $p \in P$ by the multiset definition of $\subseteq$ and we know the inhibitor arcs do not inhibit the transition by **(e)**.

We then want to show that $M_1 \overset{\delta}{\equiv} M$, i.e. $\sum_{c\in\theta} M_1(p)(c) = \sum_{c\in\theta} M(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$. Since $(M, M^\delta) \in R$ we know that $\sum_{c\in\theta} M(p)(c) = M^\delta(p)(\theta) = \sum_{c\in\theta} M_1(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$ and thus $M_1 \overset{\delta}{\equiv} M$. And since δ is stable we know that t is enabled in M.

Thus we know that the opposite direction also holds meaning that R is a bisimulation.

□

Lemma 1. *Let δ_1 and δ_2 be two partitions. Then (i) $\delta_1 \sqcup \delta_2 \geq \delta_1$ and $\delta_1 \sqcup \delta_2 \geq \delta_2$, and (ii) if δ_1 and δ_2 are stable partitions then so is $\delta_1 \sqcup \delta_2$.*

Proof. For the first part of the claim, from definition of partition union, we see that $[\theta]$ is the union of any θ' that overlaps with θ and $\delta_1 \sqcup \delta_2$ just collects all such unions for every θ. As such, it is trivial that for any $\theta \in \delta_1(p)$ there exists $\theta' \in \delta_1(p) \sqcup \delta_2(p)$ such that $\theta \subseteq \theta'$ for all $p \in P$ i.e. $\delta_1 \sqcup \delta_2 \geq \delta_1$. The same is the case for δ_2.

For the second part of the claim, let $\delta = \delta_1 \sqcup \delta_2$. Assume M and M' s.t. $M \overset{\delta}{\equiv} M'$ i.e. for all $p \in P$ and all $[\theta] \in \delta(p)$ it holds that $\sum_{c \in [\theta]} M(p)(c) = \sum_{c \in [\theta]} M'(p)(c)$ by definition of stable partition. Assume $p \in P$. From definition of partition union, we can gather that for each $[\theta] \in \delta(p)$ then for all $c, c' \in [\theta]$ there exists $c_1, ..., c_k \in [\theta]$ s.t. it holds that $c, c_1 \in \theta_1 \wedge ... \wedge c_k, c' \in \theta_k$ where $\theta_i \in \delta_1(p) \cup \delta_2(p)$ and $k \in \mathbb{N}^{>0}$ for all $i, 1 \leq i \leq k$. We use this information to show that M and M' are bisimilar. Let $M_i(p)(c_i) = \sum_{c \in [\theta]} M(p)(c)$ and $M_i(p)(c) = 0$ for all $c \in [\theta], c \neq c_i$ for some $[\theta] \in \delta(p)$. We can then create a chain $M(p) \overset{\delta_{j_1}}{\equiv} M_1(p) \overset{\delta_{j_2}}{\equiv} M_2(p) \overset{\delta_{j_3}}{\equiv} ... \overset{\delta_{j_n}}{\equiv} M'(p)$ where $j_i \in \{1, 2\}$ which implies $M(p) \overset{\delta}{\equiv} M'(p)$. The same process can then be applied to all $p \in P$ s.t. $M(p) \overset{\delta}{\equiv} M'(p)$ for all $p \in P$ meaning that $M \overset{\delta}{\equiv} M'$ by Definition 3. And since both δ_1 and δ_2 are stable both $\overset{\delta_1}{\equiv}$ and $\overset{\delta_2}{\equiv}$ are bisimulation relations implying that M and M' are bisimilar and δ is stable. □

Theorem 3. *There is a unique maximum stable partition δ s.t. $\delta \geq \delta'$ for all stable partitions δ'.*

Proof. We prove this by contradiction. Assume two maximum stable partitions δ_1 and δ_2 where $\delta_1 \neq \delta_2$. We know from Lemma 1 that $\delta_1 \sqcup \delta_2$ is stable and by Lemma 1 we know that $\delta_1 \sqcup \delta_2 \geq \delta_1$ and $\delta_1 \sqcup \delta_2 \geq \delta_2$. Thus δ_1 and δ_2 cannot both be maximum stable partitions. □

In order to compute stable partitions we need to show some properties for markings in $\mathbb{M}^{bounded}(\mathcal{N})$.

Lemma 3. *Let $\mathcal{N}$ be a CPN and δ a partition. Then for all $t \in T$ it holds that*

- **(a)** *if there exist $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \overset{\delta}{\equiv} M_2$, $M_1 \xrightarrow{t}$ and $M_2 \not\xrightarrow{t}$ then there exist $M_3, M_4 \in \mathbb{M}^{bounded}$ s.t. $M_3 \overset{\delta}{\equiv} M_4$, $M_3 \xrightarrow{t}$ and $M_4 \not\xrightarrow{t}$, and*
- **(b)** *if there exist $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ where $M_1 \overset{\delta}{\equiv} M_2$ and there exists $M_1' \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \xrightarrow{t} M_1'$ and for all $M_2' \in \mathbb{M}(\mathcal{N})$ where $M_2 \xrightarrow{t} M_2'$ it holds that $M_1' \overset{\delta}{\not\equiv} M_2'$ then there exists $M_3, M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$ where $M_3 \overset{\delta}{\equiv} M_4$ and there exists $M_3' \in \mathbb{M}^{bounded}(\mathcal{N})$ s.t. $M_3 \xrightarrow{t} M_3'$ and for all $M_4' \in \mathbb{M}^{bounded}(\mathcal{N})$ where $M_4 \xrightarrow{t} M_4'$ it holds that $M_3' \overset{\delta}{\not\equiv} M_4'$.*

Proof. Recall that $max(\mathcal{N})$ is defined as largest cardinality of all arc multisets in $\mathcal{N}$, i.e. $|W((p,t),b)| \leq max(\mathcal{N})$ for all $p \in P$ and $b \in B(t)$.

(a) Let $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \overset{\delta}{\equiv} M_2$, $M_1 \xrightarrow{t}$ and $M_2 \not\xrightarrow{t}$. We construct a marking M_3 s.t. $M_3(p) = W((p,t),b)$ for all $p \in P$ and some $b \in B(t)$ where it clearly follows that $|M_3(p)| \leq max(\mathcal{N})$ for all $p \in P$. Hence notice that $M_3 \in \mathbb{M}^{bounded}(\mathcal{N})$. We see that $M_1(p) = M_3(p) \uplus \overline{M}_3(p)$ since $M_3(p) \subseteq M_1(p)$ for all $p \in P$ where $\overline{M}_3(p)$ describes the remaining tokens in $M_1(p)$ that are not in $M_3(p)$. We know that no inhibitor arc can be the reason

M_2 is not enabled, as that would mean M_1 is not enabled either because $M_1 \overset{\delta}{\equiv} M_2$. We also know that $M_2(p) \not\subseteq W((p,t),b)$ for at least one $p \in P$ for all $b \in B(t)$.

We pick a marking M_4 where $M_4 \overset{\delta}{\equiv} M_3$, $M_4(p) \not\subseteq W((p,t),b)$ for at least one $p \in P$ and $M_2(p) = M_4(p) \uplus \overline{M}_4(p)$ for all $p \in P$ s.t. $\overline{M}_4 \overset{\delta}{\equiv} \overline{M}_3$. Notice that $M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$. We know that M_4 exists because $M_2 \overset{\delta}{\equiv} M_1$ and $M_2(p) \not\subseteq W((p,t),b)$ meaning that $M_4(p) \uplus \overline{M}_4(p) \not\subseteq W((p,t),b)$ and thus $M_4(p) \not\subseteq W((p,t),b)$ for some $p \in P$.

(*b*) Let $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \overset{\delta}{\equiv} M_2$ and $M_1' \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \xrightarrow{t} M_1'$ then for all $M_2' \in \mathbb{M}(\mathcal{N})$ where $M_2 \xrightarrow{t} M_2'$ we know that $M_1' \overset{\delta}{\not\equiv} M_2'$. We construct a marking M_3 exactly as before s.t. $M_3(p) = W((p,t),b)$ for all $p \in P$ and some $b \in B(t)$ and $M_1(p) = M_3(p) \uplus \overline{M}_3(p)$ for all $p \in P$.

We then pick a marking M_4 where $M_4 \overset{\delta}{\equiv} M_3$, $M_4(p) \subseteq W((p,t),b)$ and $M_2(p) = M_4(p) \uplus \overline{M}_4(p)$ for all $p \in P$ and $b \in B(t)$ s.t. $\overline{M}_4 \overset{\delta}{\equiv} \overline{M}_3$. We know that $M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$ since $M_4 \overset{\delta}{\equiv} M_3$. Let $M_3' \in \mathbb{M}^{bounded}(\mathcal{N})$ s.t. $M_3 \xrightarrow{t} M_3'$, which is possible because $M_3(p) \subseteq M_1(p)$ for all $p \in P$ s.t. no inhibitor arc can inhibit M_3. For the sake of contradiction now assume there exists a marking $M_4' \in \mathbb{M}^{bounded}(\mathcal{N})$ s.t. $M_4 \xrightarrow{t} M_4'$ and $M_3' \overset{\delta}{\equiv} M_4'$. Then notice that we can let $M_1'(p) = M_3'(p) \uplus \overline{M}_3(p)$ where $M_1 \xrightarrow{t} M_1'$ since $M_3(p) \subseteq M_1(p)$ for all $p \in P$ and let $M_2'(p) = M_4'(p) \uplus \overline{M_4}(p)$ where $M_2 \xrightarrow{t} M_2'$ since $M_4(p) \subseteq M_2(p)$ for all $p \in P$. But since $M_3'(p) \overset{\delta}{\equiv} M_4'(p)$ and $\overline{M}_3(p) \overset{\delta}{\equiv} \overline{M}_4(p)$ it means that $M_3'(p) \uplus \overline{M}_3(p) \overset{\delta}{\equiv} M_4'(p) \uplus \overline{M_4}(p)$ for all $p \in P$, i.e. $M_1' \overset{\delta}{\equiv} M_2'$. However, this contradicts the conditions of **(*b*)**, and as such $M_3' \overset{\delta}{\equiv} M_4'$ cannot hold. □

Theorem 4. *Given a CPN $\mathcal{N}$, the algorithm Stabilize($\mathcal{N}$) terminates and returns a stable partition of $\mathcal{N}$.*

Proof. We first prove that *Stabilize*($\mathcal{N}$) terminates. Notice that each iteration produces a new δ according to the $>$ operator, and since the operator is well-founded we know that the algorithm terminates.

We then show that for $\delta = \textit{Stabilize}(\mathcal{N})$, δ is a stable partition of $\mathcal{N}$. Recall, a partition δ is stable iff for any markings $M_1 \overset{\delta}{\equiv} M_2$ whenever $M_1 \xrightarrow{t} M_1'$ for some t and M_1' then $M_2 \xrightarrow{t} M_2'$ for some M_2' s.t. $M_1' \overset{\delta}{\equiv} M_2'$.

We prove by contradiction. Assume δ is not a stable partition. As such there must exists markings $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \overset{\delta}{\equiv} M_2$ and exist marking $M_1' \in \mathbb{M}(\mathcal{N})$ s.t. $M_1 \xrightarrow{t} M_1'$ for some transition t where for all $M_2' \in \mathbb{M}(\mathcal{N})$ s.t. $M_2 \xrightarrow{t} M_2'$ then $M_1' \overset{\delta}{\not\equiv} M_2'$.

This is exactly the property stated in the if statement on line 17 and we know from Lemma 3 that if the property is satisfied with two markings from $\mathbb{M}(\mathcal{N})$

then there exists two markings from $\mathbb{M}^{bounded}(\mathcal{N})$ that also satisfy the property. Thus the algorithm did not terminate. □

C Proofs for Section 4 (Color Approximation)

Theorem 5. *Let α be a minimum fixed point of E such that $\alpha_0(p) \subseteq \alpha(p)$ for all $p \in P$. If $M_0 \rightarrow^* M$ then $M \subseteq \alpha$.*

Proof. By induction on k we prove if $M_0 \rightarrow^k M$ then $M \subseteq \alpha$.

Base step. Firstly, in the induction basis step $k = 0$, we know that $M_0 \subseteq \alpha$ by our assumption.

Induction step. Let $M_0 \rightarrow^k M \xrightarrow{t} M'$ by some transition t with some binding $b \in B(t)$ then we want to show that $M' \subseteq \alpha$. By induction hypothesis we know that $M \subseteq \alpha$. If $M \xrightarrow{t} M'$ for some $b \in B(t)$, then $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$ for all $p \in P$. Since $E(\alpha)$ is a fixed point then $\alpha(p) = \alpha(p) \cup \mathit{set}(W((t,p),b))$ for transition t under binding b for all $p \in P$ i.e. $\mathit{set}(W((p,t),b)) \subseteq \alpha(p)$ for all $p \in P$. Thus we get $M' \subseteq \alpha$. □

D Proofs for Section 5 (Experiments)

Effect on Answered Queries In these experiments we examine which unfolding engine allows for the most query answers on their unfolded net. To allow for a fair comparison, we let each tool unfold and output the net to a PNML file. Regarding queries, both method A+B and ITS-Tools can already output the unfolded queries, but for MCC we implement our own translation from the colored queries to the unfolded queries for the given nets. For Spike we were not able to construct a query unfolder that worked consistently, for which reason Spike is excluded from these experiments.

Since we are testing the effect of the unfolding and not the verification engine, we use *verifypn* (r. 238 to include LTL queries as well) to verify the queries. We verify queries in the following categories from the Model Checking Contest: ReachabilityCardinality, ReachabilityFireability, CTLCardinality, CTLFireability, LTLCardinality and LTLFireability. There are a total of 20032 queries to be answered[1]. The results can be seen in Table 4.

We see that using method A+B to unfold nets allows for answering more queries in every category due to the generally smaller nets it unfolds to. In total we are able to answer 4.8 percentage points more queries using the unfolded nets of method A+B compared to using the unfolded nets of MCC and 5.6 percentage points more compared to ITS. It should be noted that there may have been an issue with the interaction between the queries unfolded by ITS in the LTLFireability category and the *verifypn* LTL engine, causing an explosion in the number of atomic propositions resulting in increased complexity [25]. This

[1] We disregard the LTL categories for Peterson and LamportFastMutEx as there are syntactical errors in the queries.

Cardinality Queries						
	A+B		MCC		ITS	
	Solved	%	Solved	%	Solved	%
ReachabilityCardinality	3009	88.2	2880	84.5	2894	84.9
CTLCardinality	2900	85.1	2740	80.3	2789	81.8
LTLCardinality	2778	86.8	2606	81.4	2690	84.1
Total	**8687**	**86.7**	**8226**	**82.1**	**8373**	**83.6**
Fireability Queries						
ReachabilityFireability	2664	78.2	2529	74.2	2659	78.0
CTLFireability	2546	74.7	2379	69.8	2314	67.8
LTLFireability	2471	77.2	2267	70.8	1904	59.5
Total	**7681**	**76.6**	**7175**	**71.6**	**6877**	**68.6**
Total query answers	**16368**	**81.7**	**15401**	**76.9**	**15250**	**76.1**

Table 4: Number of queries answered for the unfolded nets of each tool. The % column describes how many percent of the total available queries in each category was answered

may explain the low number of queries answered in the LTLFireability category for ITS, though we still see the effect of the smaller nets on the cardinality categories.