Unfolding of Colored Petri Nets by Color Quotienting and Approximation

Alexander Bilgram, Thomas Pedersen and Peter Taankvist

Aalborg University
Department of Computer Science, Denmark

Abstract. Colored Petri nets offer a compact and user friendly representation of traditional Petri nets also known as P/T nets. Colored Petri nets with finite color ranges can be unfolded into traditional P/T nets. However, this unfolding may produce exponentially larger P/T nets. We present two novel formal techniques based on static analyses for reducing the size of unfolded colored Petri nets. The first method identifies colors that behave equivalent in the colored Petri net and groups them into equivalence classes representing the colors, producing a smaller quotiented colored Petri net. The second method analyses which colors can ever exist in any place and excludes colors that can never be present in a given place. Both methods show great promise individually, but even more when combined. The combined method is able to reduce the size of multiple colored Petri nets from the Model Checking Contest compared to state of the art unfolders MCC, Spike and ITS-Tools, while still remaining competitive in terms of unfolding time. Lastly, we show the effect of the smaller unfolded nets by verifying queries from the 2020 Model Checking Contest using the unfolded nets from each tool.

1 Introduction

When modelling large systems it is important to verify that the system is working as intended. Since many systems are under constant change it is necessary to automate the verification process. One of the most widely used of such verifiable models are Petri nets introduced by Carl A. Petri in 1962 [26].

Petri nets are most often seen in a classical Place/Transition (P/T) form and thus most verification techniques have been developed for P/T nets [24]. However, P/T nets often become too large and incomprehensible for humans to read. Therefore, colored Petri nets (CPN) [16] have been introduced which are compressed versions of Petri nets, that employ different techniques from programming such as types, variables and logical expressions.

In CPNs each place is assigned a color domain and each token in that place has a color from the color domain. Arcs have expressions that define what colored tokens to consume and produce and transitions have guard expressions that restrict when a transition may fire.

A CPN can be translated into a P/T net if every color domain is finite through a process called *unfolding*, which allows the use of efficient verification

methods 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 the 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 that satisfies the guard is unfolded into a new transition in the unfolded net.

The size of an unfolded net can be exponentially larger than the colored net and therefore requires certain optimizations and analyses to unfold in realistic time and memory. Several types of analyses have been proposed that consider especially transition guards and arc expressions [8, 23, 27].

However, even with the optimizations there still exists CPNs that cannot be unfolded. As an example, the largest instances of the nets *FamilyReunion* [14, 7] and *DrinkVendingMachine* [13, 25] from the Model Checking Contest [21] have yet to be unfolded. This is the case since the color domains and number of possible bindings are far too large to be unfolded in realistic time and memory with current techniques. Even if the net is unfolded it may still be too large to verify due to the exponential increase in size.

We propose two new 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 the same throughout the colored net. If such colors exist in the net we can create equivalence classes that represents the colors with similar behaviour. As such, we can reduce the amount of colors that we need to consider when unfolding, because we now only have to consider the equivalence class of colors instead of each color individually.

The second method called *color approximation* detects which colors may actually 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.

Related work: Several unique approaches for unfolding CPNs effectively have been proposed. In [23] Heiner et al. analyses the arc- and guard expressions to reduce the amount of bindings that need to be considered for a given transition by collecting *patterns*. The pattern analysis is implemented in the tool Snoopy [11]. The color approximation method captures the reductions of the pattern analysis. In [27] the same authors present a technique for representing and using patterns utilizing Interval Decision Diagrams. This technique is used in the tools Snoopy [11], MARCIE [12] and Spike [5]. It proved to be generally faster than the method presented in [23].

In [8] (MCC) Dal Zilio describes a method which he calls *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. For a place to be stable it has to hold that for each transition the place is connected to, it is connected by both an input and output arc that have equivalent arc expressions. As such, it is only a syntactical check, meaning no further analysis is done. This method is especially efficient on the net BART from the model checking competition [21]. However, this method is not very general and does not find places that deviate even a little from the initial marking. The

color approximation method is a more general form of the stable places and is able to capture these deviations from the initial marking. In [8] an additional method called *Component Analysis* is presented where it is detected that a net consists of a number of copies of the same component. MCC is used in the TINA toolchain [2] and to our knowledge in the latest release of the LoLA tool [32].

GreatSPN [10] is another tool for unfolding CPNs, however in [8] it is demonstrated that MCC was able to greatly outperform GreatSPN and as such we omit GreatSPN from later experiments.

ITS-Tools [29] has an integrated unfolding engine. The tool uses a technique we refer to as *variable symmetry identification*, in which it is analyzed whether variables x and y are actually permutable in a binding. This allows for invalidation of bindings that are equivalent by constraining $x \leq y$. 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 some post-unfolding reductions that removes orphan places and transitions and removes behaviourally equivalent transitions should they make them [31].

In [20] Klostergaard presents the unfolding method implemented in *verifypn* revision 226 (untimed engine of TAPAAL [9, 15]), which is the base of our implementation. The implementation is efficient, but since it is implemented as the naive approach mentioned earlier, there are several nets which it cannot unfold such as BART.

Both the color quotienting method and the color approximation method are advanced static analyses techniques and all of the above mentioned techniques, except symmetric variables and component analysis, are captured by color approximation. Color quotienting is a completely novel technique and to our knowledge, no related work perform similar analysis on CPNs.

1.1 Bibliographical Remark

Parts of this thesis are inspired or derived from the work in our 9th semester project [22]. Specifically, the Abstract and Section 1 are modified and extended versions from the previous work where especially the related work has been extended to cover more related tools. Section 2 has been modified s.t. we instead define bisimulation and define isomorphism as a special case of bisimulation. Section 3.2 uses the same syntax and semantics as described in [22] but is described more informally in this work. Section 5 is a revised and improved version of the main contribution in [22]. Specifically, we have updated the color expansion to be a function instead of a transition relation, in order to use fixed point theory. The rest of the section is adjusted accordingly. Furthermore, we optimize the color approximation implementation presented in [22]. Lastly, Section 8.1 includes similar future work as in the previous work, but we extend it to consider new data structures and the new methods presented in this project.

2 Preliminaries

Before describing Petri nets and their colored counterparts we first define some preliminaries. We define the set of all natural numbers excluding zero as $\mathbb{N}^{>0}$ and the set of all natural numbers including zero as $\mathbb{N}^0$.

We define a Labeled Transition System (LTS) as a triple $(Q, Act, \rightarrow)$ where:

1. Q is a set of states,
2. Act is a finite, non-empty set of actions,
3. $\rightarrow \subseteq Q \times Act \times Q$ is the transition relation.

Definition 1. *(**Bisimulation**) A binary relation R over the set of states of an LTS is a bisimulation iff for $(s_1, s_2) \in R$ 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. $(s_1', s_2') \in R$*
- *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. $(s, s') \in R$. Furthermore, we say that a bisimulation R is an isomorphism if R (interpreted as a function) is a bijection. Two states s and s' are isomorphic iff there exists an isomorphism R s.t. $(s, s') \in R$.

A finite multiset over some non-empty set A is a collection of elements from A where each element occurs in the multiset a finite amount of times. Formally, a multiset S over a set A is defined as a function: $S : A \rightarrow \mathbb{N}^0$, where if $a \in A$ then $S(a)$ is the number of occurrences of element a in multiset S. We represent multisets by a formal sum:

$$\sum_{a \in A} S(a)'(a).$$

As an example assume multiset S over the set $\{x, y\}$ s.t. $S(x) = 1$ and $S(y) = 2$. This is represented as the sum $1'(x) + 2'(y)$. We denote the empty multiset over a set A by $\emptyset$ where $\emptyset(a) = 0$ for all $a \in A$. Furthermore, we denote the set of all finite multisets over A by $\mathcal{S}(A)$. We define the following multiset operations where $S, S_1, S_2 \in \mathcal{S}(A)$, $a \in A$ and $n \in \mathbb{N}^0$:

$$\begin{aligned}
&a \in S \text{ iff } S(a) > 0 && (membership)\\
&S_1 \subseteq S_2 \text{ iff } \forall a \in A.S_1(a) \leq S_2(a) && (inclusion)\\
&S_1 = S_2 \text{ iff } S_1 \subseteq S_2 \wedge S_2 \subseteq S_1 && (equality)\\
&S_1 \uplus S_2 = \textstyle\sum_{a \in A}(S_1(a) + S_2(a))'(a) && (summation)\\
&S_1 \setminus S_2 = \textstyle\sum_{a \in A}(max(0, S_1(a) - S_2(a)))'(a) && (subtraction)\\
&|S| = \textstyle\sum_{a \in A} S(a) && (cardinality)
\end{aligned}$$

The notion of multisets is based on [17].

We also introduce the notion of a color simplification. A color simplification is a way of reducing multisets of colors to sets of colors. As such we create a color simplification over a multiset S as:

$$set(S) = \{a \mid a \in S\}$$

where $set(S)$ is the set of all colors with at least one occurrence in S.

3 Colored Petri Nets

Colored Petri nets (CPN) are an extension of traditional P/T nets introduced by Kurt Jensen in 1981 [16], where places are associated with color domains and colors represent the value of tokens. Consequently, arc expressions describe what colors to consume and add to places depending on a given binding while transitions may contain guards restricting which bindings are allowed. This extension can be translated, known as *unfolding*, to a traditional P/T net if all color domains are finite as defined in [18].

There are many different definitions of CPNs from the powerful version defined in [19] that includes the ML language for modelling arcs to less powerful ones such as the one used in the Model Checking Contest [21]. We now give an abstract definition of a CPN that includes all of these definitions.

Colored Petri Net Definition

A colored Petri net is a tuple $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ (fixed for the rest of the paper) 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 non-empty set of colors,
4. $\mathbb{B}$ is a non-empty 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 $set(W((t,p),b)) \subseteq C(p)$ where $p \in P$, $t \in T$ 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. $set(M(p)) \subseteq C(p)$ where $p \in P$.

The set of all markings for a CPN is denoted $\mathbb{M}(\mathcal{N})$. To avoid the use of partial functions we allow $W((p,t),b) = W((t,p),b) = \emptyset$ and $W_I(p,t) = \infty$ for all $p \in P$ and $t \in T$, meaning that if an arc is the empty multiset then the arc never changes any marking and if an inhibitor arc is infinity it never inhibits any transition.

Notice that G, W and W_I are semantic functions. Given a binding, G and W translate to a result; a boolean for G and a multiset of colors for W.

Definition 2. ***Legal Bindings***
Let $B(t) = \{b \in \mathbb{B} \mid G(t,b) = true\}$ be the set of all bindings that satisfy the guard of transition $t \in T$.

Definition 3. ***(CPN Semantics)***
Let $\ell : T \to Act$ be a labeling function. The semantics of a CPN $\mathcal{N}$ is defined as an LTS $(\mathbb{M}(\mathcal{N}), Act, \to)$ where:

1. *$\mathbb{M}(\mathcal{N})$ is the set of states defined as all markings on $\mathcal{N}$,*
2. *Act is the set of actions, and*
3. *$\to \subseteq \mathbb{M}(\mathcal{N}) \times Act \times \mathbb{M}(\mathcal{N})$ s.t. $M \xrightarrow{a} M'$ iff there exists $t \in T$ and $b \in B(t)$ where $\ell(t) = a$ and*

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

We denote the firing of some transition $t \in T$ in marking M to 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. In order to be able to reason about automatic verification of a CPN, we need to have a finite representation of a CPN that can be passed as an input to an algorithm. One way to enforce such a finite representation is to assume that all color domains are finite and the functions W and G are effectively computable.

We then define the notion of postset and preset of transitions for some place p, given as $p^\bullet = \{t \in T \mid b \in \mathbb{B}, W((p,t),b) \neq \emptyset\}$ and ${}^\bullet p = \{t \in T \mid b \in \mathbb{B}, W((t,p),b) \neq \emptyset\}$ respectively.

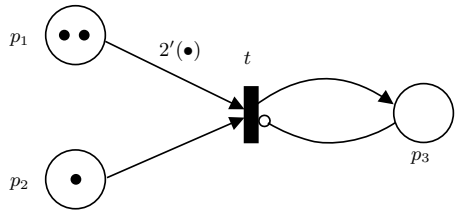
We also define the postset and and preset of places for some transition t as $t^\bullet = \{p \in P \mid b \in \mathbb{B}, W((t,p),b) \neq \emptyset\}$ and ${}^\bullet t = \{p \in P \mid b \in \mathbb{B}, W((p,t),b) \neq \emptyset\}$ respectively. Additionally, we define the preset of inhibitor arcs as ${}^\circ t = \{p \in P \mid W_I(p,t) \neq \infty\}$.

3.1 Place Transition nets

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

Figure 1a shows an example P/T net. We see places p_1, p_2 and p_3 shown as the circles in the figure. We also see a transition t displayed as a rectangle, with arrows representing the pre arcs from p_1 and p_2 and the post arc to p_3, while the arrow with a circle represents an inhibitor arc. Notice that the arc from p_1 to t consumes $2'(\bullet)$ corresponding to a weight of two. Arcs with no label have $1'(\bullet)$ as expression. Likewise, inhibitor arcs with no label have a weight of 1.

Figure 1b shows a firing of the transition t in the P/T net. Notice that we denote a marking as $p_1 : 2'(\bullet) + p_2 : 1'(\bullet)$ meaning the place p_1 has marking $2'(\bullet)$ and p_2 has $1'(\bullet)$. We do not denote places with no tokens. This notation is used in future examples.



(a) Example of P/T net. Note that the multiset of an arc without label is $1'(\bullet)$

$$p_1 : 2'(\bullet) \ + \ p_2 : 1'(\bullet) \quad \xrightarrow{t} \quad p_3 : 1'(\bullet)$$

(b) Transition firing in Figure 1a

Fig. 1: P/T net example

3.2 Integer CPN

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 ommit 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 straight forward 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:

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

where $\bowtie ::= < \mid \leq \mid > \mid \geq \mid = \mid \neq$. The semantics of guards are also straight forward and evaluate to a truth value. 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 $G(g, b_2) = \mathit{false}$.

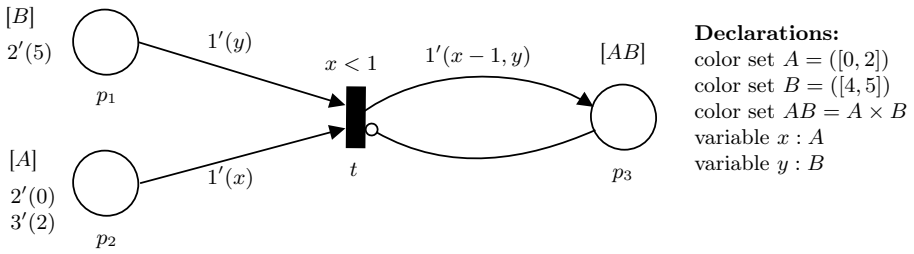
Figure 2a shows an example of an integer CPN. We see all places are associated with a set of ranges, i.e. $C(p_1) = B$ noted $[B]$. We also see that there is a guard on transition t that compares x with the integer 1. Lastly, we see that the post arc from t to p_3 creates a product of the integers x 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 2b shows an example of transition firing in Figure 2a. 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 [21] 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 2a then $A.\mathit{all} = 1'(1) + 1'(2) + 1'(3)$.

All our examples will be expressed as integer CPNs from this point forward.

3.3 Unfolding

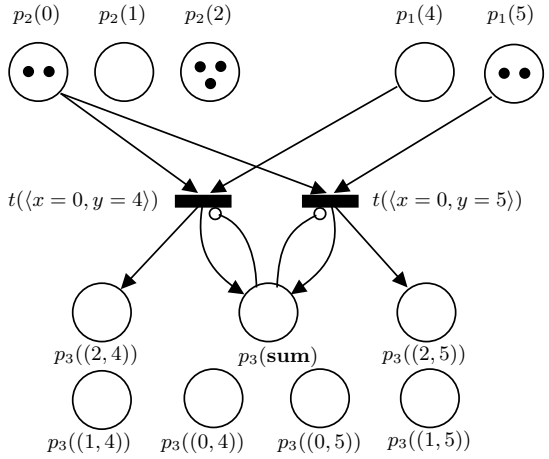
By Remark 1 we can unfold a CPN to a bisimilar P/T net since all domains are finite. Each place p is unfolded into $|C(p)|$ places, a transition is made for each legal binding and we translate the multiset of colors on the arc to a multiset of $\bullet$. For each place connected to an inhibitor arc, we create a *sum* place that contains the sum of tokens across the rest of the representative places. The *sum* place is created to ensure that inhibitor arcs functions correctly after unfolding.



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

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

(b) Transition firing in Figure 2a



(c) Unfolding of the CPN in Figure 2a to a P/T net

Fig. 2: Integer CPN and unfolding example

Definition 4. ***(Unfolding)***
Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ *be a CPN. The Petri net obtained by unfolding* $\mathcal{N}$ *is the P/T net* $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$ *where:*

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. $C^u(p) = \{\bullet\}$ *for all* $p \in P^u$,
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*
   $M_0^u(p(\boldsymbol{sum})) = |M_0(p)|'(\bullet)$

where $p(c)$ *denotes the unfolded place for color* c, $t(b)$ *denotes the unfolded transition for binding* b *and* $p(\boldsymbol{sum})$ *denotes the summation of all tokens regardless of color for place* p.

Consider the CPN in Figure 2a. The unfolded version of this is seen in Figure 2c. 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.

Theorem 1 is shown in [6, 20] but we add a small optimization on the ***sum*** places.

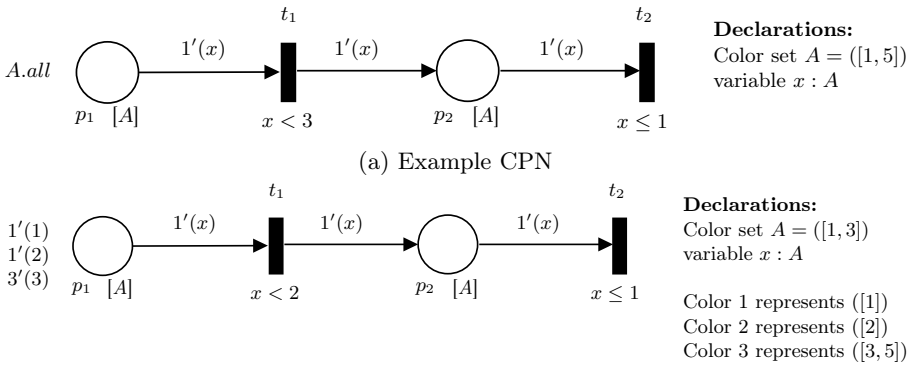
Theorem 1 ([6, 20]). *Given a CPN* $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ *and the unfolded CPN* $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$ *then* $M_0 \sim M_0^u$ *with labeling function* $\ell(t(b)) = t$ *for all* $t(b) \in T^u$.

4 Color Quotienting

Unfolding a CPN without any further analysis will often lead to many unnecessary places and transitions. As an example consider the CPN in Figure 3a, the unfolded version of this net has five places for both p_1 and p_2 and two transitions for t_1 and t_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 do see however that this is not the case for those less than 3 because of t_1 and t_2. 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 represent the CPN as a bisimilar CPN seen in 3b where the color $([3, 5])$ now represents all colors greater than or equal to 3.

Note that even if we increase the amount of colors in A to e.g. 10000 instead, we would still need the same amount of equivalence classes but only increase the amount of tokens.



(a) Example CPN

(b) Figure 3a tranlated to a bisimilar CPN with color equivalence classes

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

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

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

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

(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') where the interval $([1, 2])$ is split into two intervals, $([1])$ and $([2])$

Fig. 3: Quotienting and partition examples

We thus introduce *partitioning* over places, which is an approach where all colors with similar behaviour for a place are grouped into an *equivalence class*, denoted as θ.

Definition 5. ***(Partitioning)***
A partition δ is a function $\delta : P \to 2^{2^C} \setminus \emptyset$ 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 6. ***(Partition Marking Equivalence)***
Given a partition δ and markings M and M', we write $M(p) \overset{\delta}{\equiv} M'(p)$ for $p \in P$ 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* $p \in P$.

We now define the notion of a stable partition.

Definition 7. ***(Stable Partition)***
A partition δ *is stable if* $\overset{\delta}{\equiv}$ *is a bisimulation.*

Consider the CPN in Figure 3a. The partition shown in the Figure 3c 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 $M_1' \overset{\delta}{\not\equiv} M_2'$. On the contrary Figure 3d shows a stable partition over the same CPN. Note that we describe the partition with ranges in the same manner as with integer CPNs.

4.1 Quotienting a CPN

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

Definition 8. ***(Partition Binding Equivalence)***
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*

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

We then define classes of equivalent bindings given a δ which are sets of bindings that behave exactly the same for a given transition defined as:

$$B^\delta(t) \overset{\text{def}}{=} \{[b]_t \mid b \in B(t)\} \text{ where } [b]_t = \{b' \mid b' \overset{\delta,t}{\equiv} b\}$$

Given a stable partition we can construct a quotiented CPN where the set of colors is the equivalence classes of the stable partition, meaning that the place color function translates to a set of equivalence classes, which also requires translating the initial marking to an initial marking of equivalence classes. Likewise, the set of bindings is 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 the exact same for a transition.

Definition 9. ***(Quotiented CPN)***
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 is the CPN $\mathcal{N}^\delta = (P^\delta, T^\delta, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ where:

1. $P^\delta = P$,
2. $T^\delta = T$,
3. $\mathbb{C}^\delta = \bigcup_{p \in P} \delta(p)$
4. $\mathbb{B}^\delta = \biguplus_{t \in T} B^\delta(t)$.
5. $G^\delta(t, [b]_t) = G(t, b)$ *for all* $t \in T^\delta$ *and* $[b]_t \in B(t)$,
6. $C^\delta(p) = \delta(p)$ *for all* $p \in P^\delta$,
7. $W^\delta((p, t), [b]_t) = S$ *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* $p \in P^\delta$, $t \in T^\delta$ *and* $[b]_t \in \mathbb{B}^\delta$,
8. $W_I^\delta(p, t) = W_I(p, t)$ *and*
9. $M_0^\delta(p) = S$ *where* $S(\theta) = \sum_{c \in \theta} M_0(p)(c)$ *for all* $p \in P$ *and* $\theta \in \delta(p)$.

Note that for Item 5 and Item 7 we pick a binding b from the equivalence set of bindings $[b]_t$. However, by Definition 8 we know that it does not matter which binding from $[b]_t$ we pick as they all behave the exact same for transition t under δ.

Theorem 2. *Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$, δ be a stable partition and $\mathcal{N}^\delta = (P^\delta, T^\delta, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ be 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 9 in Definition 9. 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 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* $M^{\delta\prime} = (M^\delta(p) \setminus W^\delta((p, t), [b]_t)) \uplus W^\delta((t, p), [b]_t)$

(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

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

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:

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

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 $\theta \in \delta(p)$:

$$
\begin{array}{lll}
W^{\delta}((p,t),[b]_t)(\theta) & = & \text{Substitute by Def. 9 Item 7} \\
\sum_{c\in\theta} W((p,t),b))(c) & \leq & \text{By Equation (2)} \\
\sum_{c\in\theta} M(p)(c) & = & \text{By Equation (1)} \\
M^{\delta}(p)(\theta) & &
\end{array}
$$

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 3 since $M \xrightarrow{t} M'$. We then show that:

$$
\begin{array}{lll}
W_I^{\delta}(p,t) & = & \text{Substitute by Def. 9 Item 8} \\
W_I(p,t) & > & \text{Equation (3)} \\
|M(p)| & = & \text{Multiset Def.} \\
\sum_{c\in C(p)} M(p)(c) & = & \text{Since } (\bigcup_{\theta\in\delta(p)} \theta) = C(p) \\
\sum_{\theta\in\delta(p)} \sum_{c\in\theta} M(p)(c) & = & \text{By Equation (1)} \\
\sum_{\theta\in\delta(p)} M^{\delta}(p)(\theta) & = & \text{Multiset Def.} \\
|M^{\delta}(p)| & &
\end{array}
$$

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 $\theta \in \delta(p)$:

$$
\begin{array}{lll}
\sum_{c\in\theta} M'(p)(c) & = & \text{Substitute by Def. 3} \\
\sum_{c\in\theta}(M(p) \setminus W((p,t),b) \uplus W((t,p),b))(c) & = & \text{Substitute by multiset definitions and by enabledness of } t \\
\sum_{c\in\theta} M(p)(c) - \sum_{c\in\theta} W((p,t),b)(c) + \sum_{c\in\theta} W((t,p),b)(c) & = & \text{Substitute by Def. 9 Item 7} \\
\sum_{c\in\theta} M(p)(c) - W^{\delta}((p,t),[b]_t)(\theta) + W^{\delta}((t,p),[b]_t)(\theta) & = & \text{Since } (M, M^{\delta}) \in R \\
M^{\delta}(p)(\theta) - W^{\delta}((p,t),[b]_t)(\theta) + W^{\delta}((t,p),[b]_t)(\theta) & = & \text{by Def. 3} \\
M^{\delta\prime}(p)(\theta) & &
\end{array}
$$

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

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.

□

4.2 Computing stable partitions

With the theoretical foundation of stable partitions and quotienting of a CPN, we now show how to compute a stable partition for a given CPN. Firstly, we define a partition refinement criterion as well as a union operator on partitions.

Definition 10. ***(Partition Refinement)***
Given two partitions δ and δ' we write:

$$\delta \geq \delta' \text{ iff for all } p \in P \text{ and all } \theta' \in \delta'(p) \text{ there exists } \theta \in \delta(p) \text{ s.t. } \theta' \subseteq \theta$$

Additionally, we write $\delta > \delta'$ if $\delta \geq \delta'$ and $\delta' \neq \delta$.

As an example, consider partitions δ and δ' from Figure 3a and Figure 3d, then $\delta > \delta'$ since for all $\theta' \in \delta'(p)$ there exists $\theta \in \delta(p)$ s.t. $\theta' \subseteq \theta$ for $p \in \{p_1, p_2\}$ and $\delta \neq \delta'$.

Note that $>$ is well-founded, as for any $\delta > \delta'$ the partition δ' has more equivalence classes for some place $p \in P$ i.e. $|\delta(p)| > |\delta'(p)|$. And since the CPN is finite there can only be finitely many partitions until the smallest unique partition δ^{min} with singleton equivalence classes is reached. Note also that δ^{min} is trivially stable.

Definition 11. ***Partition Union***
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.

$$\theta \leftrightarrow \theta' \text{ iff } \theta \cap \theta' \neq \emptyset \text{ where } \theta, \theta' \in \delta_1(p) \cup \delta_2(p).$$

Let $\leftrightarrow^*$ be the transitive closure of $\leftrightarrow$. Let $[\theta] \overset{\text{def}}{=} \bigcup_{\theta' \in \delta_1(p)\cup\delta_2(p), \theta \leftrightarrow^* \theta'} \theta'$ where $\theta \in \delta_1(p) \cup \delta_2(p)$. We then define $\sqcup$ as the partition union operator s.t.

$$(\delta_1 \sqcup \delta_2)(p) = \bigcup_{\theta\in\delta_1(p)\cup\delta_2(p)} \{[\theta]\} \text{ for all } p \in P.$$

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. ***(Smallest Union)***
Given two partitions δ_1 and δ_2 then $\delta_1 \sqcup \delta_2 \geq \delta_1$ and $\delta_1 \sqcup \delta_2 \geq \delta_2$.

Proof. From Definition 11 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. □

Lemma 2. ***(Stable Union)***
Let δ_1 and δ_2 be stable partitions then $\delta_1 \sqcup \delta_2$ is also a stable partition.

Proof. 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 7.

Assume $p \in P$. From Definition 11 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 6. 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. ***(Maximum Stable Partition)***
Given a CPN $\mathcal{N}$, there is a unique maximum stable partition δ of $\mathcal{N}$ s.t. for all stable partitions δ' of $\mathcal{N}$ it holds that $\delta \geq \delta'$.

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 2 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 a stable partition we require a bounded marking set to guarantee termination. As such, we define the *max* arc size over a 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 the *max* arc size over $\mathcal{N}$ is defined as:

$$\mathbb{M}^{bounded}(\mathcal{N}) = \{M \in \mathbb{M}(\mathcal{N}) \mid |M(p)| \leq max(\mathcal{N}) \text{ for all } p \in P\}$$

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 $max(\mathcal{N})$. 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. □

We then define the procedure for computing a stable partition over some CPN shown in Algorithm 1. We start with 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 11 creates the initial partition applying the guard restrictions to the input places of the transitions. The second part from line 13 to 23 back propagates the guard restrictions throughout the net s.t. only colors that behave the same are quotiented together.

Theorem 4. *Given a CPN $\mathcal{N}$ then 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. □

4.3 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. As such, we also represent an equivalence class as a tuple of ranges as in Section 3.2. For small examples this representation may not seem important, but as some nets from the Model Checking Contest [21] have thousands of colors it becomes significant.

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

Input: $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$
Output: stable δ
let $\delta(p) := \{C(p)\}$ for all $p \in P$
for $t \in T$ **do**
  for $p \in {}^\bullet t$ **do**
    while $\exists M_1, M_2 \in \mathbb{M}^{bounded}(\mathcal{N}).M_1(p) \overset{\delta}{\equiv} M_2(p) \wedge M_1 \overset{t}{\not\rightarrow} \wedge M_2 \overset{t}{\rightarrow}$ **do**
      pick δ' s.t. $M_1(p) \overset{\delta'}{\not\equiv} M_2(p)$, $\delta > \delta'$ and $\delta(p') = \delta'(p')$ for all $p' \in P \setminus \{p\}$
      $\delta := \delta'$
    end
  end
end
let $\mathcal{Q} := P$ //Waiting list of places
while $\mathcal{Q} \neq \emptyset$ **do**
  let $p \in \mathcal{Q}$
  $\mathcal{Q} := \mathcal{Q} \setminus \{p\}$
  for $t \in {}^\bullet p$ **do**
    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**
      pick δ' s.t. $M_1 \overset{\delta'}{\not\equiv} M_2$ and $\delta > \delta'$ and $\delta'(p') = \delta(p')$ for all $p' \in P \setminus {}^\bullet t$
      $\mathcal{Q} := \mathcal{Q} \cup \{p' \mid \delta'(p') \neq \delta(p')\}$
      $\delta := \delta'$
    end
  end
end
return δ

As an example of computing stable partitions with Algorithm 1, consider the CPN in Figure 4. Table 1 shows the different stages that δ undergoes in order to become stable. From iteration 0 to 1 the guard restrictions are applied from the first for loop.

As an example of what happens in the following iterations consider iteration 1. In this iteration we pick p_3 as our place. We then look at the input places for t_1 i.e. p_1 and p_2. We see that there exist markings $M_1 = p_1 : 1'(4) + p_2 : 1'(4)$ and $M_2 = p_1 : 1'(3) + p_2 : 1'(4)$ where $M_1 \overset{\delta}{\equiv} M_2$. If we fire t_1 from the markings $M_1 \xrightarrow{t_1} M_1'$ and $M_2 \xrightarrow{t_1} M_2'$ then $M_1' = p_3 : 2'(4)$ and $M_2' = p_3 : 2'(3)$ where $M_1'(p_3) \overset{\delta}{\not\equiv} M_2'(p_3)$ since 3 and 4 are not in the same equivalence class for $\delta(p_3)$. Therefore, we let $\delta(p_1) = \{([1,3]), ([4])\}$ s.t. $M_1 \overset{\delta}{\not\equiv} M_2$.

Since the marking for p_3 is not dependent on the value of y, we do not modify $\delta(p_2)$ in this iteration. After all iterations are done a stable partition is computed.

In practice we do not iterate every bounded marking since that is inefficient. Instead we statically analyze the places, arcs and guards in order to partition

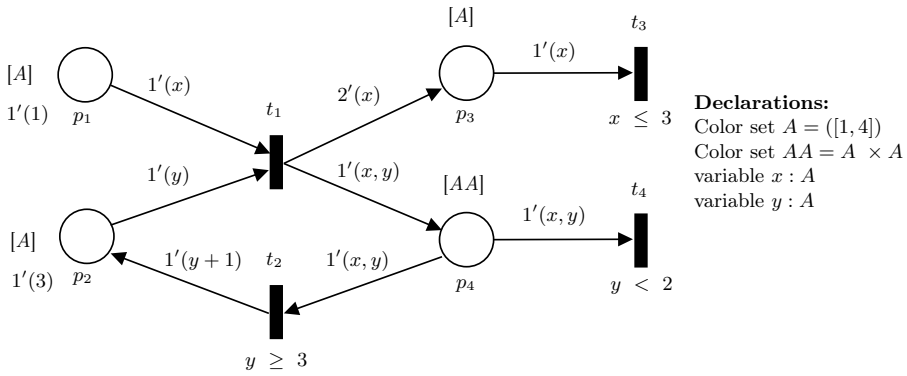


Fig. 4: Example CPN

Iteration	p_1	p_2	p_3	p_4	$\mathcal{Q}$
0	$\{([1,4])\}$	$\{([1,4])\}$	$\{([1,3]),([4])\}$	$\{([1,4]\times[1]),$ $([1,4]\times[2]),$ $([1,4]\times[3,4])\}$	$\{p_1,p_2,p_3,p_4\}$
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, $p=p_2$	-	-	-	$\{([1,4]\times[1]),$ $([1,4]\times[2]),$ $([1,4]\times[3]),$ $([1,4]\times[4])\}$	$\{p_1,p_4\}$
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 4. The 0th iteration is the state of δ just before the while loop begins. If a field has a '-' it indicates that the value is the same as the previous row.

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

Table 2: Stages of α when computing the fixed point of E for the CPN in Figure in 4. If a field has a '-' it indicates that the value is the same as the previous row.

the color sets. Each partitioning is then propagated back through the net. Additionally, we do not simply *pick* a new partition, but rather calculate equivalence classes containing as many colors as possible while satisfying the constraints of Line 7 and 18 in Algorithm 1 by statically analyzing the places, arcs and guards. As an example, in Iteration 1 of Table 1 we do not split the equivalence classes to singleton equivalence classes for p_1 but instead only split as much as is required with regards to the partitioning of p_3.

Lastly, to reduce the overhead of keeping singleton ranges we introduce a *diagonal* flag s.t. if a place is marked as diagonal we do not keep any tuples of ranges as we know each range is a singleton. In integer CPNs the flag is set if two variables are compared in a guard or if the same variable is present on two input arcs to the same transition. If a place is marked as diagonal we perform no further partitioning on the place.

5 Color Approximation

This section describes a technique for safely overapproximating what colors be present in each place of a CPN described by a *color approximation*. Essentially, we analyze which colors are reachable in each place allowing us to ignore colors that can never be present in a given place.

Definition 12. ***(Color Approximation)***
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.

Notice that $\alpha(p)$ is a complete lattice [1] for all $p \in P$ with regards to the subset inclusion. We now define an expansion of a color approximation.

Definition 13. ***(Color Expansion)***
A color expansion is a function $E : \mathbb{A} \to \mathbb{A}$ s.t.:

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

A color expansion expands the possible colors that exist in each place. Note that a color expansion only adds colors and never removes any.

Lemma 4. ***(Monotonicity)***
Let α be a color approximation then $\alpha(p) \subseteq E(\alpha)(p)$ for all $p \in P$.

Proof. For all $p \in P$ either $E(\alpha)(p) = \alpha(p) \cup \mathit{set}((W(t,p),b))$ or $E(\alpha)(p) = \alpha(p)$ and for both cases $\alpha(p) \subseteq E(\alpha)(p)$. □

Definition 14. ***(Marking Inclusion)***
Given a marking M and color approximation α, we write $M \subseteq \alpha$ iff $\mathit{set}(M(p)) \subseteq \alpha(p)$ for all $p \in P$.

Definition 15. ***(Initial Approximation)***
Let α_0 be the initial approximation where $\alpha_0(p) = set(M_0(p))$ for all $p \in P$.

Since E is a monotonic function we can compute a minimum fixed point [1] of E.

Definition 16. ***(Minimum Overapproximation)***
We say α is a minimum overapproximation iff α is a minimum fixed point of E and $\alpha_0(p) \subseteq \alpha(p)$ for all $p \in P$.

We now show that the minimum overapproximation is safe, i.e. we do not exclude any reachable colors.

Theorem 5. ***(Safeness)*** *Given a minimum overapproximation α, 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 $M_0 \subseteq \alpha$ holds trivially by Definition 16.

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 $\alpha(p) = \alpha(p) \cup set(W((t,p),b))$ for transition t under binding b for all $p \in P$ i.e. $set(W((p,t),b)) \subseteq \alpha(p)$ for all $p \in P$. Thus we get $M' \subseteq \alpha$. □

Overapproximating a CPN Given a minimum overapproximation, we can construct a safely overapproximated CPN that has possibly reduced color domains.

Definition 17. ***(Safely Overapproximated CPN)***
Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN and α be a minimum overapproximation of $\mathcal{N}$. The safely overapproximated CPN is the 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 $p \in P$.

Theorem 6. *Let $\mathcal{N}$ be a CPN and α a minimum overapproximation of $\mathcal{N}$. The reachable fragments from M_0 of the LTSs generated by $\mathcal{N}$ and $\mathcal{N}^\alpha$ are isomorphic.*

Proof. By Theorem 5 we know that for any reachable marking $M \in \mathbb{M}(\mathcal{N})$ that $M \subseteq \alpha$. Since the reachable fragments of $\mathcal{N}$ and $\mathcal{N}^\alpha$ are exactly the reachable markings we know that the reachable fragments are isomorphic. □

5.1 Computing a Minimum Overapproximation on Integer CPNs

As with color quotienting, presenting each color individually becomes inefficient when the CPNs become sufficiently large. We thus also introduce the notion of

ranges over a minumum overappoximation s.t. for a place p the approximation is a set of tuples of ranges. As an example, consider the approximation α where $\alpha(p) = \{(1,2),(2,2),(3,2),(5,6)\}$. This can be represented by the set of tuples of ranges $\{([1,3],[2]),([5],[6])\}$.

However, computing the minimum overapproximation 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. To demonstrate this we revisit the example from Figure 4. We now compute the minimum overapproximation using ranges. The full computation can be seen in Table 2. Notice that computing the minimum overapproximation using ranges may expand the number of tuples of ranges. As an example consider Iteration 4. The amount of tuples of ranges in p_2 is increased from 1 to 2.

If we instead represent the ranges of p_2 by $[1,4]$ we use at most 1 tuple of ranges to describe the approximation. This is a granularity constant, k, that can be used to tune the precision of the approximation at the trade-off of time and memory overhead by using more ranges. Essentially, k denotes how many ranges at most can be kept for any given place at any point in time. As an example assume $k = 3$ for the sets of tuples of ranges $\{([1],[1]),([2,3],[1,2]),([4,5],[2])\}$. If we change the granularity constant s.t. $k = 2$ then the ranges can be represented as $\{([1,3],[1,2])([4,5],[2])\}$. As such the color $(1,2)$ is now present in the approximation even though it was not for $k = 3$.

The value of the granularity constant is quite important depending on the net, as some nets benefit from a large value because it makes the approximation more precise and thus reducing the amount of possible bindings, while other nets benefit from a low granularity because most colors are present anyways so the high granularity only adds overhead to the computation of the minimum overapproximation. For example, for the BART nets from the Model Checking Contest [21] $k = 250$ is a good compromise, while with $k = 5$ it cannot be unfolded. Oppositely, for the BridgeAndVehicles nets, where all colors are present, $k = 5$ is good and anything larger gives a significant slowdown. For $k = 250$ it takes around 192 seconds to unfold the largest instance of BridgeAndVehicles while for $k = 5$ it takes 4.1 seconds.

6 Implementation

We implement the quotienting method presented in Section 4 and the color approximation method presented in Section 5 in C++ as extensions for the verification engine *verifypn*[15] from the TAPAAL toolchain [9].

Since both methods can potentially take a very long time depending on the net we add timeouts for both methods. For quotienting reaching the timeout means creating singleton intervals for every color in the color domain of the place. For color approximation reaching the timeout means lowering the granularity constant thus losing precision but computing faster.

We implement the notion of variable symmetry identification inspired by ITS-Tools mentioned in related work [31]. The technique works by identifying

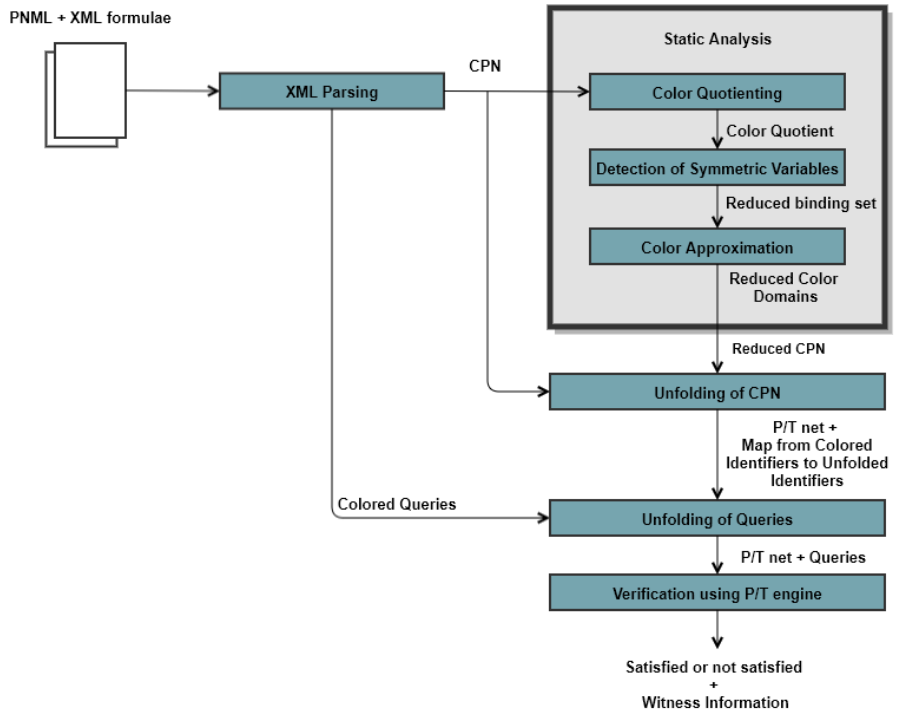


Fig. 5: Flowchart of a CPN through verifypn

whether two variables of a binding are permutable to avoid creating equivalent bindings.

For example, assume variables x, y, z with range $[1, 3]$ and an arc expression $1'(x) + 1'(y) + 1'(z)$. Assuming that there are no restrictions on these variables from guards or arcs there are $3^3 = 27$ possible bindings. However, if the variables are detected as symmetric bindings such as $\langle x = 1, y = 1, z = 2\rangle$ and $\langle x = 2, y = 1, z = 1\rangle$ are equivalent and we only need to consider 10 of the 27 bindings.

In Figure 5 a flowchart diagram of the process of unfolding and verifying a CPN can be seen. We parse the CPN, process the net with the different static analysis techniques and then use the information gained to unfold the CPN into a P/T net which can then be verified. Since the different static analysis techniques are independent any of the techniques can be disabled e.g. it is possible to only do color quotienting and color approximation and skip detection of symmetric variables. For more details on the verification process refer to [4].

6.1 Correctness of our implementation

We check the correctness of our implementation on the colored nets and queries provided in the 2020 Model Checking Contest [21]. The correctness is tested

by verifying 16 queries in each of the following categories: ReachabilityCardinality, ReachabilityFireability, CTLCardinality, CTLFireability, LTLCardinality and LTLFireability per net for 213 colored nets (of which we unfold 207). All together we test 20448 queries of which we provide an answer for 16368 queries.

We compare query answers to the Oracle database, which is a database of agreed upon answers by tools from the Model Checking Contest [30]. For the above mentioned categories there are 10096 answers where we answer consistently on all of them.

6.2 TAPAAL GUI Support

For the development of our methods and implementation we made use of the recently developed and soon to be released colored GUI of TAPAAL [28]. The GUI supports the integer CPNs described in Section 3.2 and has helped us refining and debugging our implementation.

7 Experiments

In this section we describe experiments performed with the quotienting method presented in Section 4, and the color approximation method presented in Section 5. We perform experiments between several different approaches; the quotienting method (Method A), the color approximation method (Method B), the combined quotienting, symmetric variables and color approximation method (Method A+B) against the tools MCC [8], ITSTools [29] and Spike [5]. Spike is used as a representative of MARCIE [12] and Snoopy [11], since they all use the same IDD based unfolder. As Method A and B are extensions of the TAPAAL engine *verifypn* [15], we also compare with *verifypn* revision 226 which we refer to as TAPAAL.

We compare on number of unfolded nets, size of unfolded nets, unfolding time and amount of queries answered for the unfolded nets on competition nets from the 2020 Model Checking Contest [21].

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 full repeatability package for the experiments is seen in [3].

7.1 Methodology

Size To measure the size differences we use the size ratio defined as $\frac{size_1}{size_2}$ where the size for a given net is $|P| + |T|$ where $|P|$ is the number of places and $|T|$ is the number of transitions. Since we work with division we make the following rules: $\frac{size_1}{NaN} = 0$ and $\frac{NaN}{size_2} = 1000$ where NaN is the size value for a net that has not been unfolded. Lastly, if neither can unfold the value will be 1.

Time When discussing unfolding time, we only measure pure unfolding time in seconds (and post-processing reduction time in the case of ITS), which means we exclude time used on reading, parsing, outputting etc. We make a rule that if the unfolding did not complete, the unfolding time will be set to 1000 seconds.

Cactus Plots We present the results for unfolding time and size using *cactus plots*. A cactus plot is a graph for which each method has their values sorted from lowest to highest, i.e. the first element is the one with the lowest value for a given method. As such, two points with the same x-value for different methods may not be for the same net.

Options For Method A we set the timeout to 5 seconds. For Method B we set the granularity constant to 250 and let it go down to 5 after 10 seconds.

7.2 Comparison of Tools

Number of unfolded nets Firstly, we look at how many nets each tool can unfold. This can be seen in Table 3b, where the *Total* column represents the total number of unfolded nets by the all the tools combined. The single net we cannot unfold with Method A+B is the net FamilyReunion3000 which was unfolded by MCC, though we can unfold it given 3 more minutes. In turn, Method A+B can unfold 3 nets that no other tool can unfold; DrinkVendingMachine48, 72, 96. This can be directly attributed to Method A.

Tools	**Unfolder**
TINA LoLA	MCC
ITS-Tools	ITS
TAPAAL	TAPAAL
Spike Snoopy Marcie	Spike

(a) Different tools and what unfolders they use

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

(b) Number of unfolded nets for each tool

Table 3: Summary of how many nets each tool unfolds

Size comparison In this section we illustrate the size reduction gained by the advanced analysis presented in this paper. For this we analyze size ratios between Method A+B and the other tools. In Figure 6 all size ratios where at least one comparison was not equal 1 are shown. We see that Method A+B has a smaller size ratio versus all tools for many nets, reducing some nets by more than a 1000 %. In total the size of 88 of the 208 unfolded nets are reduced compared to all other tools. Only ITS-Tools is able to sometimes unfold to a smaller net than Method A+B due to the post-reductions they employ. This is most prevalent on

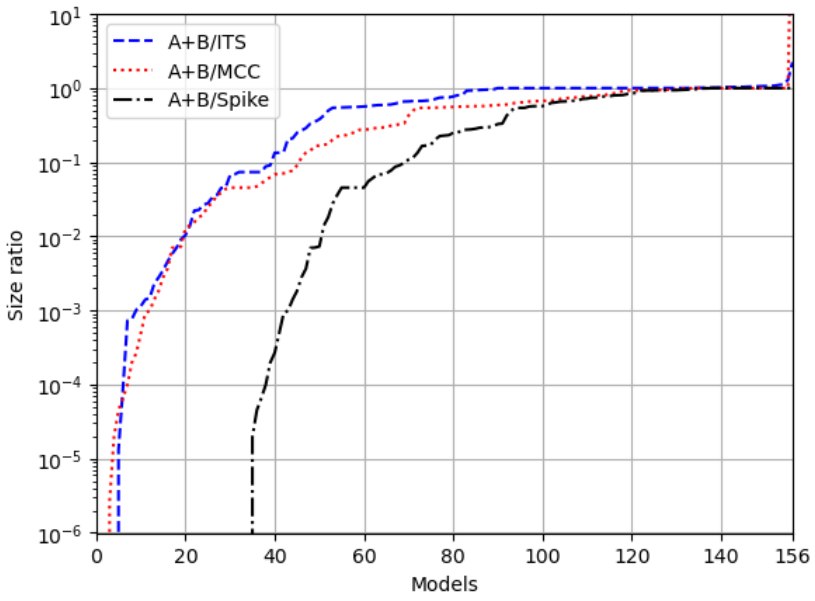


Fig. 6: Size-ratios($\frac{size_1}{size_2}$) that are not equal to 1 for all tools

the net VehicularWifi from the Model Checking Contest, which they unfold to a size of 38429 while Method A+B unfolds to a size of 85835. This is also the only net where Method A times out. Note also that the one model MCC unfolds that we do not, is seen at the end of the A+B/MCC line where it goes out of bounds.

In Table 4 the median size ratio for each type of model is presented. We see that Method A+B is better on over half of the models, and is able to reduce the size to a fraction of what the other tools can for many nets. However, we again see that ITS slightly outperforms Method A+B on a few models due to the post-reductions.

One important property of Method A+B is the ability to have constant size scaling on certain nets. This means that no matter how many extra colors are added to the color domains, the unfolded net will be the same size. As such, we can go from an increase in size with each new instance to no increase at all. The nets where this is the case can be seen in Table 5.

In general we see that Method A+B has a large effect on the sizes of the unfolded nets, and reduces the size considerably compared to the other unfolders with a few exceptions compared to ITS.

Time comparison In this section we measure the time needed to unfold. In Figure 7 the 80 slowest cases for each tool are shown. From the figure it can be

Name	A+B/ITS	A+B/MCC	A+B/Spike
DrinkVendingMachine	<0.001	<0.001	-
GlobalResAllocation	0.003	<0.001	<0.001
Referendum	0.021	0.021	0.021
Airplane	0.023	0.018	0.018
Bart	0.051	0.032	-
PermAdmissibility	0.074	0.045	0.045
DotAndBoxes	0.281	0.207	0.207
CSRepetition	0.463	0.463	0.463
Sudoku-COL-A	0.576	0.576	-
Sudoku-COL-B	0.632	0.632	-
FamilyReunion	0.686	0.685	0.686
QuasiCertifProtocol	0.947	0.947	0.947
PolyORBNT	1	0.446	0.446
PolyORBLF	1	0.557	0.557
LamportFastMutEx	1	0.807	0.807
BridgeAndVehicles	1	1	1
DatabaseWithMutEx	1	1	1
Philisophers	1	1	1
PhilisophersDyn	1	1	1
TokenRing	1	1	1
SharedMemory	1.007	1	1
SafeBus	1.009	0.997	0.997
NeoElection	1.04	0.057	0.077
Peterson	1.047	1	1
VehicularWifi	2.234	0.174	-

Table 4: Size-ratios on the median models for all tools. A '-' indicates that the median net was not unfolded by the tool

seen that ITS, MCC and Method A+B are close in performance and follow the same development, while Spike is slower. We see that ITS is generally fast on the nets that are unfolded in less than 10 seconds, however it becomes slower and has problems unfolding the larger nets. We see that MCC and Method A+B are very similar in performance, but Method A+B edges ahead on the largest nets. Overall we see that our advanced analyses with Method A+B adds very little overhead, but decreases the size of the unfolded nets significantly compared to the other unfolders.

Effect on Queries Answered 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.

	A+B	ITS	MCC	Spike
AirplaneLD-COL-0010	55	145	177	177
AirplaneLD-COL-0020	55	265	327	327
AirplaneLD-COL-0050	55	625	777	777
AirplaneLD-COL-0100	55	1225	1527	1527
AirplaneLD-COL-0200	55	2425	3027	3027
AirplaneLD-COL-0500	55	6025	7527	7527
AirplaneLD-COL-1000	55	12025	15027	15027
AirplaneLD-COL-2000	55	24025	30027	30027
AirplaneLD-COL-4000	55	48025	60027	60027
BART-COL-002	447	668	1410	-
BART-COL-005	447	1670	3117	-
BART-COL-010	447	3340	5962	-
BART-COL-020	447	6680	11652	-
BART-COL-030	447	12509	17342	-
BART-COL-040	447	13360	23032	-
BART-COL-050	447	16700	28722	-
BART-COL-060	447	20040	34412	-
DrinkVendingMachine-COL-02	22	76	96	96
DrinkVendingMachine-COL-10	22	28780	111280	111280
DrinkVendingMachine-COL-16	22	248352	1118752	1118752
DrinkVendingMachine-COL-24	22	1685232	8309232	-
DrinkVendingMachine-COL-48	22	-	-	-
DrinkVendingMachine-COL-76	22	-	-	-
DrinkVendingMachine-COL-98	22	-	-	-
Referendum-COL-0010	7	52	52	52
Referendum-COL-0015	7	77	77	77
Referendum-COL-0020	7	102	102	102
Referendum-COL-0050	7	252	252	252
Referendum-COL-0100	7	502	502	502
Referendum-COL-0200	7	1002	1002	1002
Referendum-COL-0500	7	2502	2502	2502
Referendum-COL-1000	7	5002	5002	5002

Table 5: Nets with constant size scaling. A '-' indicates that the net was not unfolded by the tool

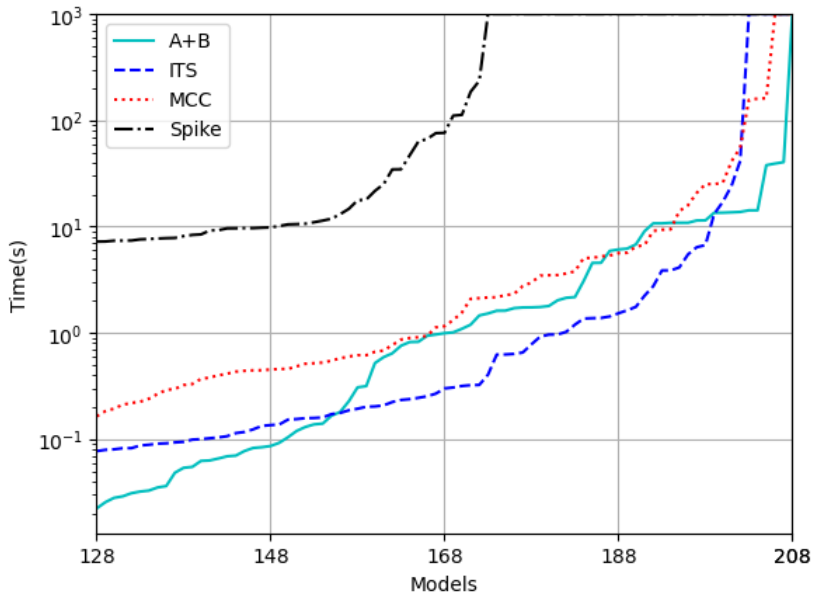


Fig. 7: Worst 80 cases for the unfolding time

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 6.

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 [31]. This might explain the low number of queries answered in the LTLFireabilty category for ITS, though we still see the effect of the smaller nets on the cardinality categories.

[1] We disregard the LTL catagories 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 6: 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

8 Conclusion

We presented two independent methods, color quotienting and color approximation, for translating colored Petri nets (CPN) into reduced CPNs. The first method analyzed the CPN in order to create equivalence classes of colors that contain colors that behave the same throughout the net. We used these equivalence classes to construct new symbolic colors for the net thus reducing the amount of bindings which in turn reduces the size of the unfolded Place/Transition net. We proved that this method produces a quotiented CPN that is bisimilar to the original CPN.

The second method once again analysed the CPN in order to construct a color approximation that described the possible colors that can exist in any given place. This allowed us to ignore colors that can never exist thus removing possible bindings and reducing the size of the unfolded net. We proved that this method produces a safely overapproximated CPN that is isomorphic to the original CPN.

We implemented the methods in the tool *verifypn* which is part of the TAPAAL tool family, as an extension of the naive unfolding method already implemented.

Experimental results showed that both the methods greatly increase the number of nets from the Model Checking Contest that can be unfolded compared to the naive approach. Furthermore, the methods in conjunction unfold more nets than either of the methods individually. Comparing with other state of the art unfolding tools the methods in conjunction unfolds 88 of 208 CPNs to a smaller Place/Transition nets and in 51 cases to a net that is under half the size compared to the competition, while still remaining competitive on time used.

Lastly, we performed experiments on the unfolded nets showing that the smaller nets allowed for more cardinality and fireability query answers than any other tool with 4.8 percentage points more query answers than the second best tool.

8.1 Future Work

There are multiple interesting prospects of colored Petri net unfolding for future development.

Firstly, for the combined color quotienting and color approximation method we currently only apply each computation once i.e. we first compute the quotiented CPN and then the safely overapproximated CPN. It may be interesting to continuously pass the CPN back and forth between the two methods as this may decrease the size of the unfolded net even further.

To speed up the stable partition algorithm it is a possibility to create a data structure that keeps track of what places are dependent of what places, thus when a change is made it is immediately possible to know which places are affected. This may be more efficient than using the CPN structure directly.

Instead of only using the static analysis for unfolding, we can also use it for answering queries directly. For example, we can tell just from the color approximation of a given net whether a place can ever receive tokens of any color. Additionally, we are able to tell if a transition t can fire by checking if the color approximation of each of the output places of t is empty. This can be used to simplify the query prior to the unfolding, which is interesting to investigate further, to see if it will reduce the verification time.

To further reduce the size of the net and speed up the unfolding process, structural reduction directly on the colored Petri net can be explored. For example, on the model FamilyReunion from the Model Checking Contest we see potential reductions. In particular in combination with the methods presented in this paper, as removed places and transitions can have a big impact on the equivalence classes of the CPN as well as the colors that may exist.

8.2 Acknowledgements

We would like to thank the following creators of other unfolders. Yann Thierry-Mieg for his answers and modifications concerning the unfolder used in ITS-tools [29]. Additionally, we would like to thank him for providing new options for outputting the unfolded net in PNML format and the unfolded queries in XML format. Silvano Dal Zilio for his answers concerning the MCC unfolding tool and updating his tool such that it outputs a dictionary from colored place/transition names to unfolded place/transition names which could be used to unfold queries. Monika Heiner and Christian Rohr for their answers concerning the tools Snoopie, Marcie and Spike.

Lastly, we would like to thank Jiří Srba for his excellent supervision and many sparring sessions, especially for his willingness to help at any time of the day.

We would also like to thank Peter Gjøl Jensen for his supervision and technical advising on the implementation.

Bibliography

[1] L. Aceto, A. Ingolfsdottir, K.G. Larsen, and J. Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.

[2] 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 - INT J PROD RES*, 42:2741–2756, 2004. doi.org/10.1080/00207540412331312688.

[3] A. Bilgram, T. Pedersen, and P. Taankvist. Unfolding-experiments-repeatabilitypackage. https://github.com/Marglib/Unfolding-Experiments-RepeatabilityPackage, 2021.

[4] F. Bønneland, J. Dyhr, P. G. Jensen, M. Johannsen, and J. Srba. Simplification of ctl formulae for efficient model checking of petri nets. In V. Khomenko and O. H. Roux, editors, *Application and Theory of Petri Nets and Concurrency*, pages 143–163, Cham, 2018. Springer International Publishing. http://doi.org/10.1007/978-3-319-91268-4_8.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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*, page 492–497. Springer-Verlag, 2012. doi.org/10.1007/978-3-642-28756-5_36.

[10] E. Gilberto Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis. 30 Years of GreatSPN. 7927:227–254, 2016. doi.org/10.1007/978-3-319-30599-8_9.

[11] 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.

[12] 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.

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

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

[15] 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. doi.org/10.1007%2F978-3-662-53401-4_16.

[16] 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.

[17] K. Jensen. Coloured Petri nets: A high level language for system design and analysis. *Advances in Petri Nets 1990*, pages 342–416, 1991. https://doi.org/10.1007/3-540-53863-1_31.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] A. B. Kristensen, T. Pedersen, and P. Taankvist. Efficient Unfolding of Colored Petri Nets by Color Overapproximation. 9th semester project report, Department of Computer Science, Aalborg University, 2020. https://projekter.aau.dk/projekter/files/403056910/Efficient_Unfolding_of_Colored_Petri_Nets_by_Color_Overapproximation.pdf.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] TAPAAL Team. cpn-gui-dev branch, 2021. https://code.launchpad.net/~tapaal-contributor/tapaal/cpn-gui-dev.

[29] 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.

[30] Y. Thierry-Mieg. pnmcc-models-2020. github.com/yanntm/pnmcc-models-2020, 2020.

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

[32] Karsten 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.