On Deciding Behavioral Properties for Petri Nets:

Timed-Arc Petri Nets and their Extensions

Ragnhildur Óskarsdóttir
Sigmar Stefánsson
Tómas Jónasson

Master Thesis

Software System Engineering
Department of Computer Science
Aalborg University, Denmark

13. May 2005

Acknowledgement

We wish to take the opportunity to thank our supervisor Jiří Srba for his guidance and support. His knowledge and encouragement has, without doubt, greatly improved this work.

Aalborg, 13. May 2005

Ragnhildur Óskarsdóttir

Sigmar Stefánsson

Tómas Jónasson

Abstract

The Petri net model is well known in the field of model checking and behavioral properties of it have been well studied in recent years. Properties such as reachability, coverability, deadlock and liveness have all been shown to be decidable for this model. However the Petri net model lacks the ability to model real time systems and therefore, a timed extension of it, the timed-arc Petri net model, was introduced. However, decidability of all the behavioral properties mentioned above has not been studied in timed-arc Petri nets to the same extent as in regular Petri nets.

In this paper we give results for decidability of all of these properties for the timed-arc Petri net model, as well as other extensions of that model, defined completely by us or inspired by the work of others.

Contents

1 Introduction

When developing a system it is often desirable to know if it behaves correctly (i.e. behaves according to the system specification) and effectively to solve the task it is intended to solve, before the actual construction starts. A convenient way to analyze the effectiveness of a system is to build a model of it so that its structure and dynamic behavior can be studied beforehand. A model is a representation, often in mathematical terms, of what are felt to be the important features of the object or system under study [16]. Through modeling, knowledge about the modeled system can be obtained without the inconvenience and cost of manipulating the actual phenomenon, and suggestions for improvements and changes can be made.

Model checking is a formal verification technique based on the exploration of the states of a model. Model checking can often be automated and various tools have been developed to do so, all having the same principle, checking if a model of a system satisfies a given property. The tools accept the system specification (a model) and some property specification that the final system is expected to satisfy, and through automatic verification the tools output yes if the given model satisfies the property, otherwise generate a counter example (Figure 1).

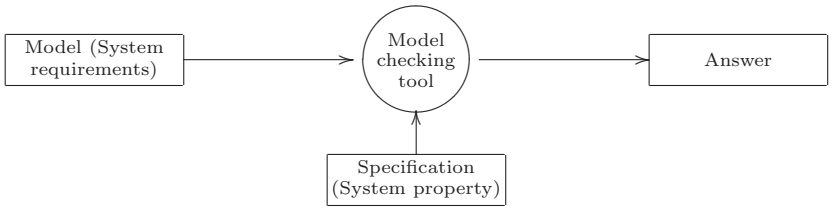


Figure 1: Model checking

A major current challenge in automatic program verification is to extend model checking to transition systems with infinite state space. In this paper we focus on a time extension of the Petri net model, namely the timed-arc Petri net model, define extensions of it and look at decidability questions concerning some of their behavioral properties.

A Petri net (PN) consists of two kinds of nodes, called places and transitions, where transitions are fired to move tokens around in the net, from one place to another through connecting arcs, according to given rules. Tokens in a Petri net represent resources of the modeled system. The location of all tokens in a Petri net is called a marking, where we state how many tokens occupy each place. A Petri net is an infinite-state system since it has an infinite control part, namely markings.

The Petri net model has been extended to handle real time systems, by introducing time aspects to it. We look at an extension called timed-arc Petri nets (TAPNs), where tokens are assigned a value from the TAPN's time domain

that represents its age, i.e. how many time units have elapsed since the token was created. Time intervals are also introduced on the arcs in a timed-arc Petri net and they limit what tokens can be used to fire a transition.

Reachability, coverability, deadlock and liveness are some of the most common behavioral properties checked for when studying Petri nets. Reachability is known to be decidable for the Petri net model [13] but undecidable for the timed-arc Petri net model [18]. Coverability on the other hand is decidable for TAPNs, and in [3] an algorithm that uses backwards reachability and existential zones (EZ) to decide it, is presented. Deadlock and liveness have been shown to be decidable for Petri nets [6] but to our knowledge these two properties have not been closely studied for TAPNs, with regard to decidability and undecidability.

Our Contribution
We give two polynomial reductions for behavioral properties of timed-arc Petri nets, one from reachability to deadlock and another from deadlock to liveness. We define three different types of intervals for timed-arc Petri nets (open, half open and closed) and modify the technique of backward reachability with existential zones (the EZ algorithm) [3] so it can be used to decide coverability for all timed-arc Petri nets, regardless what type of intervals they have.

For distributed timed-arc Petri nets (DTAPNs) [15] with discrete time domain we give a reduction to regular Petri nets preserving timed-reachability, and for DTAPNs with continuous time domain we modify the existential zones algorithm so it can be used to decide coverability for them.

We define age-preserving TAPNs, show how to construct a regular TAPN with discrete time domain that simulates a given age-preserving TAPN, also with discrete time domain, preserving strong bisimilarity. We also show how to modify the EZ algorithm so it can be used to decide coverability for age-preserving TAPNs with continuous time domain. We then define TAPNs with inhibitor arcs and finally TAPNs with reset arcs.

Related Work
The distributed timed-arc Petri net model we define is inspired by the original definition from [15]. The authors examine some behavioral properties for those, and related, nets including the decidability of coverability for DTAPNs. We, on the other hand, modify an algorithm used to define coverability for TAPNs so it can be used in the same way for DTAPNs.

The backward reachability analysis is a technique that has been applied to many classes of systems. In [1] the authors use this approach to decide coverability for well-structured systems. In [3, 4] the authors apply this technique, combined with existential zones, to a special class of systems called better-quasi ordered transition systems, one of which is the timed-arc Petri net model. We use this technique to show decidability of coverability for our extensions of the TAPN model, modifying it every time for it to work for each extension.

Polynomial time reductions for Petri nets are the given in [6] and [9], but we do not know of any similar work which has been done for timed-arc Petri nets.

Outline
This paper is structured as follows. The next section presents some preliminaries for this paper including a section on labeled transition systems. In Section 3 we define both the Petri Net and the timed-arc Petri net models, present our classification of TAPNs, define both models as labeled transition systems and state the behavioral properties which are of interest to us, for both models. We present our undecidability results for TAPNs in Section 4. In Section 5 we introduce the technique we use when deciding coverability for TAPNs namely backwards reachability analysis using existential zones, show how we extend it to handle all types of intervals mentioned earlier and give an example of how it works. Finally we define four extensions of the TAPN model, take a look at whether or not we can reduce them to timed-arc Petri nets, and modify the technique described in Section 5 so it can be used to decide coverability for those extensions which do not turn out to be Turing powerful.

2 Preliminaries

This section is intended to list the notation used in later chapters and state or recall several definitions. Notations are presented in a way suitable for this instead of in their most general form.

2.1 On the Number Theory

We use $\mathbb{N}$ to denote the set of all natural numbers including 0, i.e. $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\mathbb{Z}$ to denote the set of all integers, i.e. $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ and $\mathbb{R}^+$ to denote the set of all non-negative real numbers.

Notation:
The infinity symbol ∞ is a positive infinite quantity that is greater than every number, i.e. $\forall r \in \mathbb{R}$, $r < \infty$. We assume the following, $\forall r \in \mathbb{R}$:

$$\infty + r = \infty$$

$$\infty - r = \infty$$

2.2 On the Theory of Multisets

Definition 2.1.
*A **multiset** M is a function from an element set A to the natural numbers, giving the multiplicity of each element, $M : A \to \mathbb{N}$.*
A multiset is finite if $\sum_{a \in A} M(a) < \infty$.

Given two multisets M_1 and M_2 over a set A, the union of them, $M_1 \cup M_2$, is defined as follows:

$$(M_1 \cup M_2)(a) = M_1(a) + M_2(a) \quad \forall a \in A$$

The intersection of two multisets, $M_1 \cap M_2$ is defined as follows:

$$(M_1 \cap M_2)(a) = min\{M_1(a), M_2(a)\} \quad \forall a \in A$$

where *min* returns the smallest element in a set.

Definition 2.2.
*We define the **set of all finite multisets** $A^{\oplus}$ over a set A to be*

$$A^{\oplus} = \{M : A \to \mathbb{N} \mid \sum_{a \in A} M(a) < \infty\}.$$

Typically, a multiset is written as a set of ordered pairs, e.g. a multiset where $M(a) = 3$ and $M(b) = 2$ is written $\{(a, 3), (b, 2)\}$. We will sometimes write out the full multiset, i.e. $\{a, a, a, b, b\}$.

Definition 2.3.
Given a multiset A over $\mathcal{D}$ and a value $d \in \mathcal{D}$ we define an operator $\lessdot\!+$ that adds the value d to every element in A s.t.

$$A \lessdot\!+ d = \{a + d \mid a \in A\}$$

2.3 Labeled Transition Systems

Labeled transitions systems capture the idea behind process behavior. A process is an agent that exists in a given state, it can perform an action communicating with the environment and then becomes another process.

A labeled transition system represents these processes as nodes in labeled directed graphs.

Definition 2.4.
*A **labeled transition system** $\mathcal{L}$ is a triple $(S, Act, \rightarrow)$, where*

- *S is a possibly infinite set of states*
- *Act is a finite set of labels s.t. $S \cap (Act) = \emptyset$, where each label represents an action observable from the environment*
- *$\rightarrow \subseteq S \times Act \times S$ is the transition relation*

Transition $(s, \alpha, s') \in \rightarrow$ is denoted as $s \xrightarrow{\alpha} s'$ and intuitively means that the system can move from a state s to a state s' while performing the observable action α. $s \rightarrow s'$ denotes that there exists an action $\alpha \in Act$ such that $s \xrightarrow{\alpha} s'$, and $\xrightarrow{*}$ denotes the reflexive and transitive closure of $\rightarrow$.

As an example, we look at a labeled transition system with two states, s_1 and s_2. From s_1 we can do an a transition and reach state s_2, from which we can only do a b transition and return to state s_2. This LTS is shown in Figure 2.

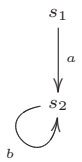


Figure 2: Labeled transition system

2.3.1 Comparing LTSs

To compare two LTSs we mention a few notions of equivalences. We start with isomorphism:

Definition 2.5.
Two transition systems $\mathcal{L} = (S, Act, \rightarrow)$ *and* $\mathcal{L}' = (S', Act, \rightarrow')$ *are* ***isomorphic****, written* $\mathcal{L} \cong \mathcal{L}'$*, if there exists a bijective function (onto and one-to-one)* $\hbar : S \rightarrow S'$ *s.t. for all* $\alpha \in Act$ *and any states* s_1, s_2 *in* S *and* s'_1, s'_2 *in* S'*, where* $\hbar(s_1) = s'_1$ *and* $\hbar(s_2) = s'_2$*, the following holds:*

$$s_1 \xrightarrow{\alpha} s_2 \text{ if and only if } s'_1 \xrightarrow{\alpha} s'_2.$$

Let us have a look at the example in Figure 3 that shows two isomorphic LTSs.

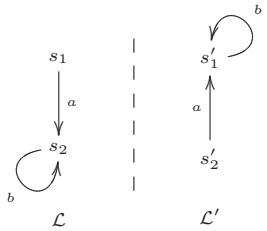


Figure 3: Isomorphic LTSs

Let $\hbar(s_1) = s'_2$ and $\hbar(s_2) = s'_1$. From state s_1 we can do an a transition and reach the state s_2 in $\mathcal{L}$. The same holds for $\hbar(s_1) = s'_2$, where we can do an a transition and reach $\hbar(s_2) = s'_1$ in $\mathcal{L}'$. From state s_2 we can do a b transition and become s_2 in $\mathcal{L}$. The same holds for $\hbar(s_2) = s'_1$ in $\mathcal{L}'$. Thus, $\mathcal{L} \cong \mathcal{L}'$.

A weaker equivalence notion is that of bisimilarity. A bisimulation is a binary relation on the set of states in a labeled transition system.

Definition 2.6.
Let $\mathcal{L} = (S, Act, \rightarrow)$ *be a labeled transition system. A binary relation* $\mathcal{R} \subseteq S \times S$ *is a* ***(strong) bisimulation*** *if and only if whenever* $(s_1, s'_1) \in \mathcal{R}$ *then for each* $\alpha \in Act$*:*

- *if* $s_1 \xrightarrow{\alpha} s_2$ *then* $s'_1 \xrightarrow{\alpha} s'_2$ *for some* $s'_2 \in S$ *s.t.* $(s_2, s'_2) \in \mathcal{R}$
- *if* $s'_1 \xrightarrow{\alpha} s'_2$ *then* $s_1 \xrightarrow{\alpha} s_2$ *for some* $s_2 \in S$ *s.t.* $(s_2, s'_2) \in \mathcal{R}$.

Definition 2.7.
Two states s_1 *and* s'_1 *in transition systems* $\mathcal{L} = (S, Act, \rightarrow)$ *and* $\mathcal{L}' = (S', Act', \rightarrow')$ *respectively, are said to be* ***(strongly) bisimilar****, written* $s_1 \sim s'_1$*, if there exists a strong bisimulation* $\mathcal{R}$ *s.t.* $(s_1, s'_1) \in \mathcal{R}$.

For example, the LTSs $\mathcal{L}$ and $\mathcal{L}'$ in Figure 4 are strongly bisimilar where:

$$\mathcal{R} = \{(s_1, s'_1), (s_2, s'_2), (s_2, s'_3)\}$$

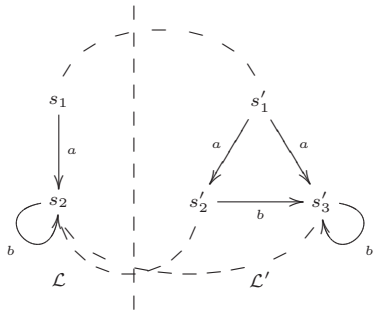


Figure 4: Strongly bisimilar LTSs

The weakest notion of equivalence that we will mention is weak bisimulation. The intuition of weak bisimulation is that it abstracts away the internal behavior of the system by introducing a special transition τ to model internal or silent actions, and add it to the set of allowed transitions of the transition system. Then we can define $\stackrel{\alpha}{\Rightarrow} \subseteq S \times S$ for all $\alpha \in Act$ to be the relation $\stackrel{\alpha}{\longrightarrow}$ preceded and followed by arbitrary many τ actions.

Definition 2.8.
We define $\Rightarrow \subseteq S \times B \times Act \times B \times S$ *to be a transition relation on a labeled transition system* $\mathcal{L} = (S, Act, \rightarrow)$*, where*

$$B = \begin{cases} \emptyset \\ \{\tau_1, \ldots, \tau_n\} \quad \textit{where } n \in \mathbb{N} \end{cases}$$

Transitions $(s, \tau, \alpha, \tau, s') \in \Rightarrow$, $(s, \alpha, \tau, \tau, s') \in \Rightarrow$ and $(s, \alpha, s') \in \Rightarrow$ are all denoted by $s \stackrel{\alpha}{\Rightarrow} s'$. Intuitively that means that the system can move from a state s to a state s' while performing arbitrary many internal τ actions before and after the observable action α. $s \Rightarrow s'$ denotes that there exists an action $\alpha \in Act$ such that $s \stackrel{\alpha}{\Rightarrow} s'$, and $\stackrel{*}{\Rightarrow}$ denotes the reflexive and transitive closure of $\Rightarrow$.

Definition 2.9.
Let $\mathcal{L} = (S, Act, \rightarrow)$ *be a labeled transition system. A binary relation* $\mathcal{R} \subseteq S \times S$ *is a* ***weak bisimulation*** *if and only if whenever* $(s_1, s'_1) \in R$ *then, for each* $\alpha \in Act$*:*

- *if* $s_1 \stackrel{\alpha}{\longrightarrow} s_2$ *then* $s'_1 \stackrel{\alpha}{\Rightarrow} s'_2$ *for some* $s'_2 \in S$ *s.t.* $(s_2, s'_2) \in \mathcal{R}$
- *if* $s'_1 \stackrel{\alpha}{\longrightarrow} s'_2$ *then* $s_1 \stackrel{\alpha}{\Rightarrow} s_2$ *for some* $s_2 \in S$ *s.t.* $(s_2, s'_2) \in \mathcal{R}$.

Definition 2.10.
Two states s_1*,* s'_1 *in transition systems* $\mathcal{L} = (S, Act, \rightarrow)$ *and* $\mathcal{L}' = (S', Act', \rightarrow)$

respectively, are said to be ***weakly bisimilar****, written* $s_1 \approx s'_1$*, if there exists a binary relation* $\mathcal{R} \subseteq S \times S'$ *which is a weak bisimulation s.t.* $(s_1, s'_1) \in \mathcal{R}$*.*

As an example, the LTSs $\mathcal{L}$ and $\mathcal{L}'$ in Figure 5 are weakly bisimilar where:

$$\mathcal{R} = \{(s_1, s'_1), (s_2, s'_1), (s_3, s'_2)\}$$

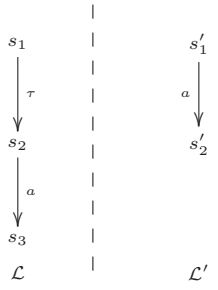


Figure 5: Weakly bisimilar LTSs

It is obvious that given two states s and t in a transition system $\mathcal{L}$

$$s \cong t \;\Rightarrow\; s \sim t \;\Rightarrow\; s \approx t.$$

We will use these notions of equivalence later on to compare different classes of timed-arc Petri nets and extensions.

3 Petri Nets

The theory of Petri nets was proposed and formally defined by Carl Adam Petri in his doctoral dissertation in 1962. Petri nets are one of the oldest and most studied formalisms for the investigation of concurrency [7]. Murata [14] gives a detailed overview of the history of Petri nets until 1989, and further details on the developments of the Petri net theory can be found in [11].

3.1 The Petri Net Model

A Petri net is a directed, bipartite graph consisting of two kinds of nodes called *places* and *transitions*. *Arcs* are placed either from a place to a transition or from a transition to a place.

Definition 3.1.
*A **Petri net (PN)** is a 4-tuple $N = (P, T, In, Out)$ where:*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places*
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions s.t.* $P \cap T = \emptyset$
- $In : P \times T \rightarrow \mathbb{N}$ *is a function that associates a natural number to each pair* (p, t)
- $Out : T \times P \rightarrow \mathbb{N}$ *is a function that associates a natural number to each pair* (t, p)

A place p is called an *input place* of transition t if $In(p,t) \neq 0$, and an *output place* of t if $Out(t,p) \neq 0$. We use the notation ${}^\bullet t = \{p \mid In(p,t) \neq 0\}$ to denote the multiset of all input places of transition t and similarly we use $t^\bullet = \{p \mid Out(t,p) \neq 0\}$ to denote the multiset of all output places of t.

In a graphical representation of a Petri net, we draw places as circles, transitions as boxes and arcs are drawn as arrows between them. The value of $In(p,t)$ states how many arcs are drawn from place p to transition t and the value of $Out(t,p)$ states how many arcs are between t and p. Figure 6 shows an example of how a Petri net can model a classical Producer-Consumer system, where one part of the system (the producer) produces some item and transports it to a common buffer, while another part of the system (the consumer) can fetch these items and consume them.

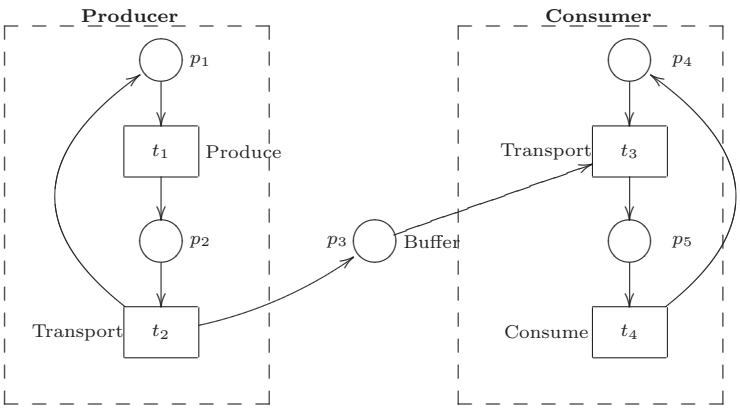


Figure 6: Example of a Petri net

Tokens are used to represent resources in a Petri net. They are graphically represented as a dot inside a place, one for each token positioned there. A *marking* assigns a nonnegative number of tokens, k, to each place p and we say that p *is marked with* k *tokens.*

Definition 3.2.
A ***marking*** *on a TAPN* $N = (P, T, In, Out)$ *is a function* $M : P \to \mathbb{N}$*, where* $M(p)$ *represents the number of tokens in the place* $p \in P$*.*

We sometimes write a marking as a vector $M = (x_1, x_2, ..., x_n)$ where n is the total number of places and the i-th component represents the number of tokens in the place p_i, assuming that $P = \{p_1, p_2, \ldots, p_n\}$.

Definition 3.3.
A ***marked Petri net (MPN)*** *is a pair* (N, M_0) *where* M_0 *is the initial marking on a Petri net* N*.*

When a system has been modeled by a Petri net, its dynamic behavior is simulated by *firing* an *enabled* transition to change the marking of the net.

Definition 3.4.
Let $N = (P, T, In, Out)$ *be a Petri net,* M *a marking on it and* $t \in T$*. We say that* t *is* ***enabled*** *at marking* M *if and only if:*

$$M(p) \geq In(p, t) \ \forall p \in {}^\bullet t$$

i.e. we have at least $In(p, t)$ *tokens in every input place* p *of* t*.*

By firing transition t *we reach a marking* M' *(written* $M[t\rangle M'$*) derived according to the* ***firing rule****:*

$$M'(p) = M(p) - In(p, t) + Out(t, p) \ \forall p \in P$$

So a transition t is enabled if every input place p of t has at least as many tokens as the input arcs leading from p to t. When t fires, it consumes as many tokens, from each of its input places, as the number of input arcs from p to t and creates as many tokens in each output place p as the number of output arcs leading from t to p.

In Figure 7 we have a marked version of the net in Figure 6, with the initial marking $M_0 = (0, 1, 0, 1, 0)$, and in Figure 8 we see how firing transition t_2 makes the net reach the marking $M' = (1, 0, 1, 1, 0)$.

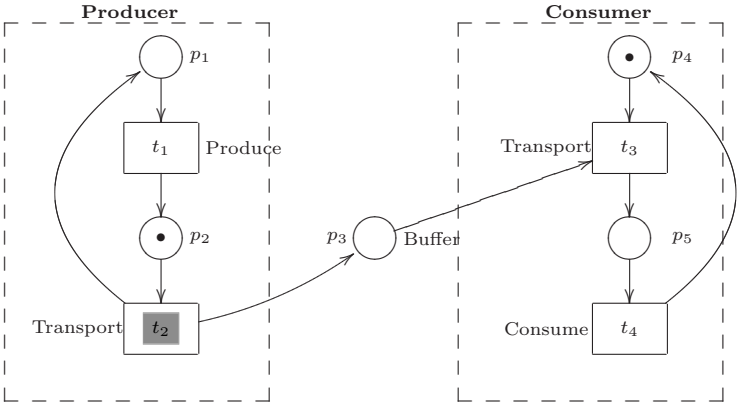


Figure 7: Marked Petri net

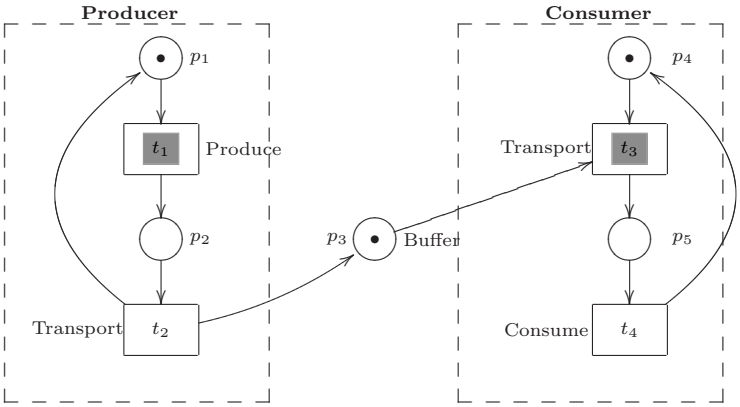


Figure 8: Transition firing

Notice that an enabled transition may or may not fire, and if we have more than one transition enabled at a given time, it is non-deterministically chosen, which of them fires.

Definition 3.5.
*A **firing sequence** is a sequence of transition firings $\sigma = t_1 t_2 ... t_n$ leading from a marking M to a marking M' s.t. $M[t_1\rangle M_1[t_2\rangle M_2 ... M_{n-1}[t_n\rangle M'$. $M[\sigma\rangle M'$ is an equivalent notation. Notice that if σ is empty then $M[\sigma\rangle M$.*

Definition 3.6.
*A marking M' is said to be **reachable** from a marking M if there exists a firing sequence σ such that $M[\sigma\rangle M'$.*

Definition 3.7.
*The **reachability set** of a Petri net N, $R(N, M_0)$, is the set of all reachable markings from the initial marking M_0.*

The reachability set of a single marking M on N, $R(N, M)$ (which is not an initial marking), is the set of all reachable markings from M.

3.1.1 PN as Labeled Transition System

Petri nets can be represented as labeled transition systems.

Definition 3.8.
*A **labeled PN** is a 6-tuple $N = (P, T, In, Out, \Lambda, \lambda)$ where (P, T, In, Out) is a PN and*

- *Λ is finite set of labels*
- *$\lambda : T \to \Lambda$ is a labeling function that gives each transition $t \in T$ a label*

Definition 3.9.
*We define $\mathcal{L}(N) = (S, Act, \to)$ as the **labeled transition system** generated by a labeled PN $N = (P, T, In, Out, \Lambda, \lambda)$ s.t.*

- *$S = [P \to \mathbb{N}]$ is a set of states (markings in N)*
- *$Act = \Lambda$ is the set of labels*
- *$\to \subseteq S \times Act \times S$, is the set of allowed transitions defined by:*
  - *$M \xrightarrow{a} M'$ if and only if $\exists t \in T$ s.t. $M[t\rangle M'$ and $a = \lambda(t)$*

Let us have a look at an example. Given the PN N in Figure 9 we generate the LTS $\mathcal{L}(N)$, a part of which is shown in Figure 10.

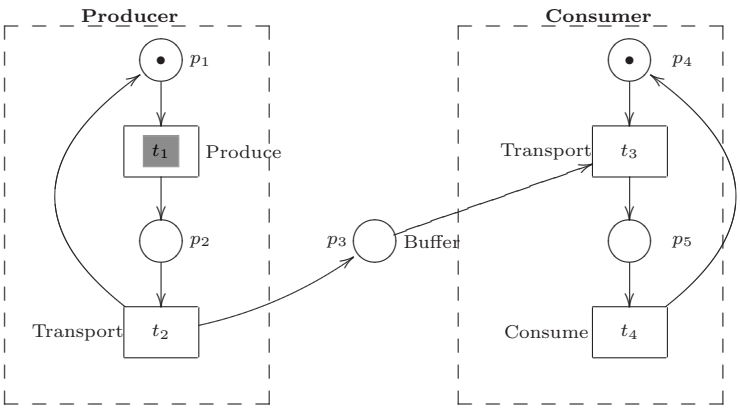


Figure 9: Petri net N with initial marking $M_0 = (1, 0, 0, 1, 0)$

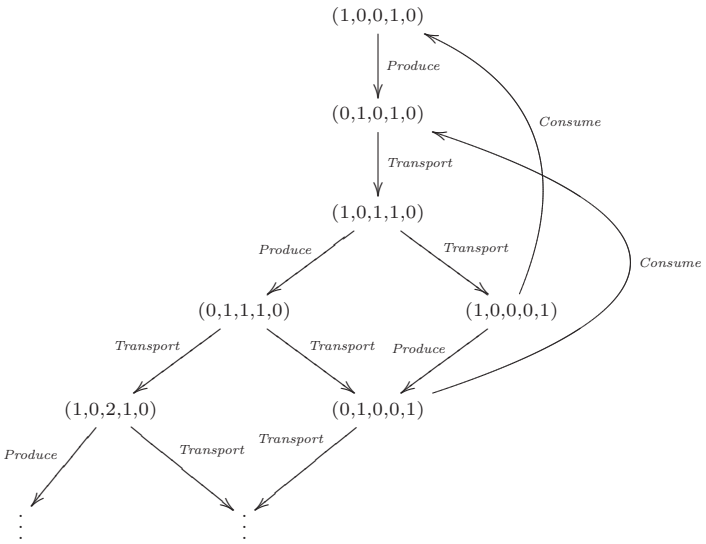


Figure 10: The LTS $\mathcal{L}(N)$

Notice that transitions t_2 and t_3 have the same label "Transport" in $\mathcal{L}(N)$. Since two transitions in the PN have the same label, we do not care about which one of them fires, as long as a "Transport" action is observed from the environment.

3.1.2 Behavioral Properties

When we have modeled a system with a Petri net, we can check various properties of the modeled system by verifying that the Petri net has certain behavioral

properties. Here we define and illustrate some interesting properties of Petri nets that have been considered in the literature.

Definitions

Reachability
By looking at a marking of a Petri net, we can verify if it is possible to reach another marking by firing a sequence of transitions.

Definition 3.10.
*The **reachability** problem for a marking M' in a marked Petri net (N, M_0), is the question of whether $M' \in R(N, M_0)$.*

Reachability is useful when checking if a system can possibly reach a certain state. For example, we can check if a system ever reaches a state where some desired behavior can or has occurred.

It is well known that reachability is decidable for Petri nets [13].

Coverability
Coverability can be regarded as relaxed reachability. The coverability question asks if we can reach a marking that has at least some given number of tokens in some given places.

Definition 3.11.
*A marking M' on a Petri net N **covers** a marking M (written $M \leq M'$) if:*

$$M(p) \leq M'(p) \ \forall p \in P$$

A marking M' covers a marking M if there are equal or more tokens present in every place of the net in marking M' than in marking M.

Definition 3.12.
*The **coverability** problem for marked Petri net (N, M_0) and a given marking M, is the question whether or not we can reach a marking M' that covers M from the initial marking, i.e. if $\exists M'$ s.t. $M' \in R(N, M_0)$ and $M \leq M'$.*

Following these definitions, we state the following lemma:

Lemma 3.1.
If a marking M' on a Petri net N covers a marking M on N then $R(N, M) \subseteq R(N, M')$, i.e. the reachability sets are monotone.

Coverability is known to be decidable for Petri nets [16].

Deadlock
If a Petri net can reach some marking from which it can not do anything, it deadlocks.

Definition 3.13.
*For a Petri net $N = (P, T, In, Out)$ a transition $t \in T$ is **potentially fireable** in a marking M, if $\exists M' \in R(N, M)$ and t is enabled in M'.*

Thus, a potentially fireable transition is a transition which can eventually become enabled.

Definition 3.14.
A Petri net $N = (P, T, In, Out)$, with initial marking M_0, can ***deadlock****, if $\exists M \in R(N, M_0)$ s.t. no transition is potentially fireable from M.*

It can be very useful for us to be able to tell whether or not some Petri net can deadlock, because very often, that is something we want to avoid in our systems.

Reachability was shown to be polynomially reducible to deadlock for PNs in [6], proving that deadlock is decidable for the model. Later these two properties were shown to be recursively equivalent [9].

Liveness
Live Petri nets never deadlock, and moreover all transitions in a live Petri net can eventually become enabled.

Definition 3.15.
A Petri net $N = (P, T, In, Out)$, with initial marking M_0, is ***live****, if all transitions $t \in T$ are potentially fireable $\forall M \in R(N, M_0)$.*

Liveness is usually a desirable property of Petri nets. By correctly modeling a system with a Petri net and verifying that it is live, it is guaranteed that the modeled system will never deadlock.

In [12] the author showed that liveness is recursively equivalent to reachability, proving that liveness is decidable. Later, the deadlock problem was proved to be polynomially reducible to the liveness problem [6] but finding a polynomial time reduction from reachability to liveness remains an open problem.

3.2 The Timed-Arc Petri Net Model

The basic Petri net model is good for modeling some systems, but when it comes to time dependant behavior, the basic model is insufficient. Several extended models have been proposed that take time features into account (for a survey see [5, 21]). As an example, we have the timed-transition Petri net model where time is associated with transitions in the net, implying the duration of the transition firing [8]. Another example is the timed-place Petri net model where time is associated with places in the net, implying that a transition only becomes enabled if tokens have been present in its input places for some time units [19]. We will however focus on yet another timed extension of the basic Petri net model called *timed-arc Petri nets (TAPN)*.

In timed-arc Petri nets we introduce two time related attributes. First each token in the net is annotated with a non-negative value, representing its age. The second attribute are time intervals which are assigned to each arc in the net. We will refer to arcs leading from a place to a transition as *input arcs* of that transition, and similarly we will refer to arcs leading from transitions to

places as *output arcs*. As in regular Petri nets, a transition can only fire when all its input places have at least one token but furthermore, the ages of the tokens need to lie within the intervals of the corresponding input arcs. When a transition is fired, the ages of the tokens created in the output places are non-deterministically chosen to be a value within the interval of the corresponding output arc.

Let us look at the formal definitions. Before we define timed-arc Petri nets, we provide a formal definition of the intervals that are associated with each arc in the TAPN.

Definition 3.16.
We define sets of ***intervals****. A single interval can be written in five different ways,* $[a, a]$*,* $[a, b]$*,* $(a, b]$*,* $[a, b)$ *or* (a, b)*, where* $a \in \mathbb{N}$ *and* $b \in \mathbb{N} \cup \{\infty\}$ $(a < b)$*, are the lower and upper limits of the interval respectively.*

- $x \in [a, a]$ *denotes* $x = a$
- $x \in [a, b]$ *denotes* $a \leq x \leq b$
- $x \in [a, b)$ *denotes* $a \leq x < b$
- $x \in (a, b]$ *denotes* $a < x \leq b$
- $x \in (a, b)$ *denotes that* $a < x < b$

We call $[a, b]$ *closed intervals,* $[a, b)$ *and* $(a, b]$ *half open intervals, and* (a, b) *open intervals.*

We then define three sets of intervals:

- $Interv_1$ *is a set intervals defined by the following abstract syntax*
  $\mathcal{I}_1 ::= [a, b]$
- $Interv_2$ *is a set intervals defined by the following abstract syntax*
  $\mathcal{I}_2 ::= [a, b] \mid [a, b) \mid (a, b]$
- $Interv_3$ *is a set intervals defined by the following abstract syntax*
  $\mathcal{I}_3 ::= [a, b] \mid [a, b) \mid (a, b] \mid (a, b)$

The following relation holds for these three sets: $Interv_1 \subseteq Interv_2 \subseteq Interv_3$ *where* $Interv^{\oplus}$ *is the set of finite multisets over* $Interv$*.*

Notation:
We use $[\![a, b]\!]$ as a general term when using open, half open or closed intervals, e.g. $[\![a, b]\!] \in \{[a, b], (a, b), [a, b), (a, b]\}$.

Before we formally define the timed-arc Petri net model we fix the interval set $Interv \in \{Interv_1, Interv_2, Interv_3\}$.

Definition 3.17.
A ***timed-arc Petri net (TAPN)*** *is a 4-tuple* $N = (P, T, In, Out)$ *where*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places*

- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions s.t.* $P \cap T = \emptyset$
- $In : P \times T \to Interv^{\oplus}$ *is a function that associates to each input arc* (p, t) *a finite multiset of intervals*
- $Out : T \times P \to Interv^{\oplus}$ *is a function that associates to each output arc* (t, p) *a finite multiset of intervals*

Definition 3.18.
For a TAPN $N = (P, T, In, Out)$*, the multiset of* ***input arcs*** *of transition* $t \in T$ *is* $In(t) = \{(p, \mathcal{I}) | \mathcal{I} \in In(p, t)\}$ *and similarly the multiset of* ***output arcs*** *of transition* $t \in T$ *is* $Out(t) = \{(p, \mathcal{J}) | \mathcal{J} \in Out(t, p)\}$*.*

We use the notation ${}^\bullet t = \{p \mid In(p, t) \neq \emptyset\}$ to denote the multiset of all input places of transition t and similarly we use $t^\bullet = \{p \mid Out(t, p) \neq \emptyset\}$ to denote the multiset of all output places of t.

The time domain $\mathcal{D}$ indicates how time elapses in the model. We consider discrete and continuous time elapsing. We keep the following definitions independent of which time domain applies, and fix the time domain $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ before we look at markings in TAPNs.

Definition 3.19.
A ***marking*** M *on a TAPN* $N = (P, T, In, Out)$ *is a function* $M : P \to \mathcal{D}^{\oplus}$*.*

In a marking each place is annotated with a certain number of tokens and each token has an age from the time domain $\mathcal{D}$.

We write a marking M in two different ways. One is $M = \{(p_1, 2), (p_1, 4.3), (p_1, 4.3)\}$, where each pair consists of the name of the place and a number from the time domain $\mathcal{D}$ representing the age of a token positioned in this particular place. The other way is to write $M(p_1) = \{2, 4.3, 4.3\}$, giving us the tokens positioned in place p_1.

Definition 3.20.
Marked timed-arc Petri net (MTAPN) *is a pair* (N, M_0)*, where* N *is a TAPN and* M_0 *is an initial marking on* N*, s.t.*

$$\forall p \in P.\ \forall x \in M_0(p).\ x = 0$$

So as initial markings on a TAPN N, we only allow tokens of age 0.

Before we go any further, let us take a look at an example of a timed-arc Petri net. Earlier we have seen a classical Producer-Consumer system (Figure 6) modeled as a regular Petri net but now we look at a more specific case:

> The producer is a dairy that produces bottles of milk and sends it to a supermarket, and the consumer is a customer. The dairy produces fresh milk and sends it to the supermarket, a process which takes 0 to 1 days (depending on locations). The supermarket has a certain

capacity so it can only handle a certain amount of milk bottles. As time goes by the milk gets older and expires after 7 days and should then be discarded from the supermarket. Customers buy bottles of milk that have not expired and take them home where they can consume it when they want to.

We can model this easily with the timed-arc Petri net in Figure 11. Note that in this paper we sometimes draw timed-arc Petri nets where no interval is assigned to some arcs. These arcs should be considered to have the interval $[0, \infty]$.

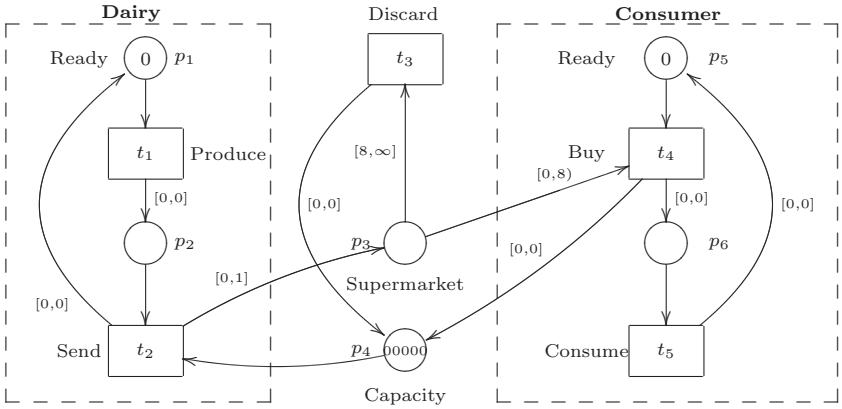


Figure 11: Dairy example

We represent bottles of milk as tokens in the net. A supermarket is represented as a buffer where tokens can be collected and grow older as time goes by. The capacity place in the net makes sure that only certain number of tokens can exist in the buffer at each time, i.e. in our example the supermarket can only hold 5 bottles of milk in its store at each time. Time intervals on arcs in the Petri net, limit both the consumption and creation of tokens. For example, the arc from p_1 to t_1 has the interval $[0, \infty]$ which represents that transition t_1 is enabled, no matter how old the token in place p_1 is, where again the transition from p_3 to t_4 that has interval $[0, 8)$, implies that the transition t_4 is enabled only if there is a token in p_3 of an age from 0 to less than 8 (and we also require a token of any age from p_5). In a graphical representation of timed-arc Petri nets such as this, each token is symbolized in the net by its age (e.g. the initial marking in Figure 11 has two tokens of age 0 in places p_1, and p_5 and 5 tokens in p_4 representing the capacity of the buffer, all of age 0 as well).

Now we have defined the TAPN model. We define how we fire transitions and how we model the passage of time.

Definition 3.21.

Let $N = (P, T, In, Out)$ be a TAPN, M a marking on it and $t \in T$.

*We say that t is **enabled** at marking M if and only if:*

- $\forall p \in {}^\bullet t.\ \exists X_p = \{x_1, \ldots, x_n\} \subseteq M(p)$, *where* $In(p,t) = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, *s.t.* $x_i \in \mathcal{I}_i$ *for* $1 \leq i \leq n$

i.e. in each input place, for every input arc, there exists at least one token whose age fits the interval of that arc.

The set of tokens created, when t is fired, is any set

- $Y_p = \{y_1, \ldots, y_m\}\ \forall p \in t^\bullet$, *where* $Out(t,p) = \{\mathcal{J}_1, \ldots, \mathcal{J}_m\}$, *s.t.* $y_i \in \mathcal{J}_i$ *for* $1 \leq i \leq m$

i.e. in each output place, for every output arc, one token, whose age fits the interval of that arc, will be created.

Tokens are only removed from input places of t and only created in output places of t. So we define:

$$X_p = \emptyset\ \forall p \in (P \setminus {}^\bullet t)$$
$$Y_p = \emptyset\ \forall p \in (P \setminus t^\bullet)$$

*Now we can define the **firing rule** formally.*

$$M'(p) = (M(p) \setminus X_p) \cup Y_p\ \forall p \in P$$

As for the basic PN model, when a transition is fired, the marking of the net changes according to the given *firing rule*. The firing rule for TAPNs states that when transition t fires we remove a set of tokens, whose ages are within the intervals associated to the input arcs, from each input place of t, and add a set of new tokens to each output place of t, where the ages of the tokens are within the intervals associated to the output arcs. Again, the firing of a transition t is denoted by $M[t\rangle M'$ where M and M' are markings.

An enabled transition t can fire, but it does not have to. We model the passage of time which is simply a different kind of a transition function. The firing of a transition that lets time elapse for $d \in \mathcal{D}$ time units is denoted by $M[\epsilon(d)\rangle M'$ where M and M' are markings.

Definition 3.22.
*Let (N, M_0) be a MTAPN, where M and M' are markings on it, and $N = (P, T, In, Out)$. We write a **time-elapsing transition** as $M[\epsilon(d)\rangle M'$ for $d \in \mathcal{D}$ whenever $M' = M + d$, s.t. $M'(p) = M(p) \lessdot + d$*

For example if $M_0[\epsilon(1.36)\rangle M'$ then M' is a new marking where the age of all tokens in the marking M_0 have been raised by 1.36, but the number of tokens and their positions remain the same.

Definition 3.23.
*A **firing sequence** for a MTAPN is a sequence of transition firings* $\sigma = \sigma_1\sigma_2...\sigma_n$ *leading from a marking* M *to a marking* M' *s.t.* $M[\sigma_1\rangle M_1[\sigma_2\rangle M_2 \ldots M_{n-1}[\sigma_n\rangle M'$*, where* σ_i $(1 \leq i \leq n)$ *is either a transition firing (t) or a time elapsing transition* $(\epsilon(d))$*. An equivalent notation is* $M[\sigma\rangle M'$*. Notice that, if* σ *is empty,* $M[\sigma\rangle M$*.*

Both the definition of a reachable marking (Definition 3.10) and the reachability set (Definition 3.7) are the same as for regular Petri nets, using the new definition of a firing sequence.

Let us now look at an example of how transitions become enabled, how they are fired and how time passes in our TAPN in Figure 11.

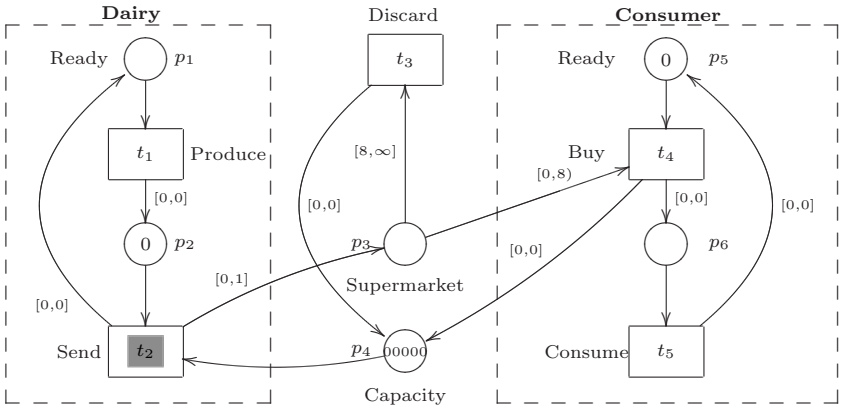


Figure 12: Dairy example - enabled transitions

Figure 12 shows us how transition t_2 is enabled since there are tokens in places p_2 and p_4. By firing that transition we reach the marking shown in Figure 13, where we have a new token of age 0 in place p_3 and transitions t_4 and t_1 are enabled. If we fire neither t_4 nor t_1, time can elapse until we reach the marking in Figure 14 where transition t_3 is now enabled instead of t_4, since the token in place p_3 is now of age 8.

3.2.1 Classes of TAPNs

In our definition of the timed-arc Petri net model we have used a very general way of defining the time attributes of the model, both the types of intervals and the time domain. Later on we will look at and compare TAPNs with the same structure but different time attributes. For this purpose we define classes of TAPNs.

Definition 3.24.
Let $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ *be a time-domain and* $Interv \in \{Interv_1, Interv_2, Interv_3\}$

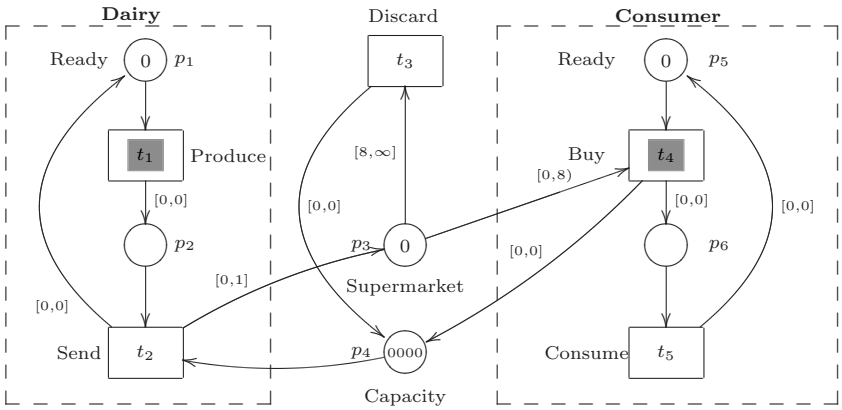


Figure 13: Dairy example - transition fires

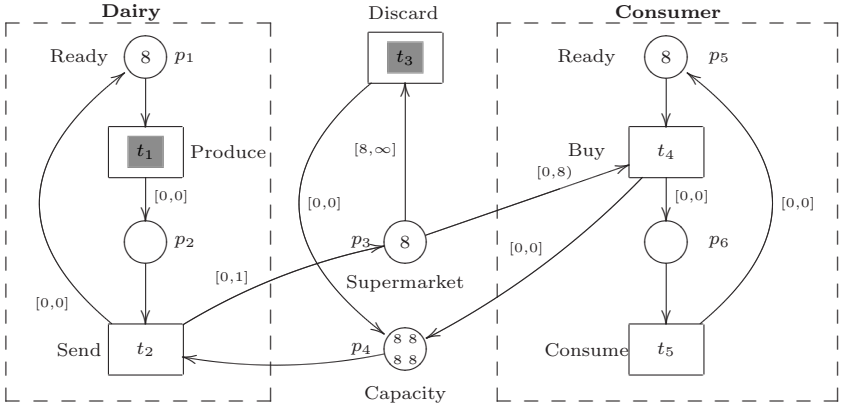


Figure 14: Dairy example - time elapses

be the set of allowed intervals. Then we define $\mathcal{N}(\mathcal{D}, Interv)$ *as the* ***class of all TAPNs*** *with time domain* $\mathcal{D}$ *and interval set* $Interv$*.*

Remark
Given a net in every class we can see that if $Interv \in \{Interv_1, Interv_2, Interv_3\}$ and $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$:

- $\mathcal{N}(\mathbb{N}, Interv_1)$ denotes a regular TAPN in discrete time, allowing only closed intervals.
- $\mathcal{N}(\mathbb{N}, Interv_2)$ denotes a TAPN in discrete time, allowing half open and closed intervals.

- $\mathcal{N}(\mathbb{N}, Interv_3)$ denotes a TAPN in discrete time, where all types (closed, half open and open) of intervals are allowed.
- $\mathcal{N}(\mathbb{R}^+, Interv_1)$ denotes a regular TAPN in continuous time, allowing only closed intervals.
- $\mathcal{N}(\mathbb{R}^+, Interv_2)$ denotes a TAPN in continuous time, allowing half open and closed intervals.
- $\mathcal{N}(\mathbb{R}^+, Interv_3)$ denotes a TAPN in continuous time, where all types (closed, half open and open) of intervals are allowed.

We show that a TAPN N belonging to the class $\mathcal{N}(\mathbb{N}, Interv_3)$ can be reduced to a net N' in $\mathcal{N}(\mathbb{N}, Interv_1)$ preserving isomorphism.

The intervals in a net in $\mathcal{N}(\mathbb{N}, Interv_3)$ are open, i.e. the endpoints are not included in the interval. By definition, for a token to fit in an open interval (a, b), its age has to be greater than a and less than b.

If the net in question is using continuous time it means that the token fits in an open interval as soon as its age becomes a fraction greater than the lower limit a and until it is just a fraction less than the upper limit b. If the net on the other hand is using discrete time a token still does not fit the interval if it is exactly of the age a and when it gets older it jumps straight to the age $a+1$. Therefore the lower limit a of an open interval can always be changed to $a+1$ in a closed interval without changing the behavior of the net, when reducing a TAPN in $\mathcal{N}(\mathbb{N}, Interv_3)$ to a TAPN in $\mathcal{N}(\mathbb{N}, Interv_1)$.

Similarly for the upper limit of a open interval, when using a continuous time the token fits the interval until it is exactly of the age b. But if the TAPN is using discrete time the token fits at age $b-1$ but at age b it does not fit. Therefore, as for the lower limit, the upper limit b of an open interval can be changed to the closed interval $b-1$ when reducing a TAPN in $\mathcal{N}(\mathbb{N}, Interv_3)$ to a TAPN in $\mathcal{N}(\mathbb{N}, Interv_1)$ without changing any behavior. The only exception from this is if the upper limit of an open interval is ∞, then the value does not change when the reduction is made.

We start by defining a function that performs the operations described above on a set of intervals. The function $l : Interv \to \mathbb{N}$ returns the lower bound of a given interval, increased by one if appropriate:

$$l(\mathcal{K}) = \begin{cases} a & \text{if } \mathcal{K} = [a, b] \text{ or } \mathcal{K} = [a, b) \\ a+1 & \text{if } \mathcal{K} = (a, b] \text{ or } \mathcal{K} = (a, b) \end{cases}$$

Similarly, the function $u : Interv \to \mathbb{N}$ returns the upper bound of a given interval, decreased by one if appropriate:

$$u(\mathcal{K}) = \begin{cases} b & \text{if } \mathcal{K} = [a, b] \text{ or } \mathcal{K} = (a, b] \\ b-1 & \text{if } \mathcal{K} = [a, b) \text{ or } \mathcal{K} = (a, b) \end{cases}$$

Given a set $\mathcal{X} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ of intervals, let $bound(\mathcal{X})$ be the set:

$$bound(\mathcal{X}) = \{[l(\mathcal{K}), u(\mathcal{K})] \mid \mathcal{K} \in \mathcal{X} \text{ and } l(\mathcal{K}) \leq u(\mathcal{K})\}$$

Recall that $\infty - n = \infty \; \forall n \in \mathbb{N}$.

Lemma 3.2.
All intervals in bound($\mathcal{X}$) are in $Interv_1$.

Example 3.1.
Given a set of intervals $\mathcal{X} = \{[1,3],(1,3],[1,3),(1,3),(2,3),[3,\infty)\}$,

$$bound(\mathcal{X}) = \{[1,3],[2,3],[1,2],[2,2],[3,\infty]\}$$

Lemma 3.3 follows:

Lemma 3.3.
For any $x \in \mathbb{N}$, the following is true:

- $x \in [a,b] \Leftrightarrow x \in [a,b]$
- $x \in (a,b] \Leftrightarrow x \in [a+1,b]$
- $x \in [a,b) \Leftrightarrow x \in [a,b-1]$
- $x \in (a,b) \Leftrightarrow x \in [a+1,b-1]$

Now we can formally give the reduction:

Theorem 3.1.
Given a TAPN N in $\mathcal{N}(\mathbb{N}, Interv_3)$ we can construct in linear time, an isomorphic TAPN N' in $\mathcal{N}(\mathbb{N}, Interv_1)$, s.t. $\mathcal{L}(N) \cong \mathcal{L}(N')$.

Proof.
Given $N = (P,T,In,Out)$ in $\mathcal{N}(\mathbb{N}, Interv_3)$ we construct $N' = (P',T',In',Out')$ in $\mathcal{N}(\mathbb{N}, Interv_1)$ s.t.

- $P' = P$
- $T' = T$
- $In'(p,t) = bound(In(p,t)) \; \forall p \in P. \forall t \in T$
- $Out'(t,p) = bound(Out(t,p)) \; \forall p \in P. \forall t \in T$

We show that $\mathcal{L}(N) \cong \mathcal{L}(N')$:

N and N' have the same set of possible markings since $P = P'$. Thus, we know that the set of all possible markings $[P \to \mathbb{N}^{\oplus}]$ (states) in $\mathcal{L}(N)$ and in $\mathcal{L}(N')$ is the same, and the bijective function is the identity function:

$$\hbar(M) = M \quad \forall M \in [P \to \mathbb{N}^{\oplus}]$$

Due to Lemma 3.3, given any two markings M_1 and M_2 in $\mathcal{L}(N)$ and any $\alpha \in \Lambda \cup \{\epsilon(n)|n \in \mathbb{N}\}$:

$$\textit{if } M_1 \xrightarrow{\alpha} M_2 \textit{ in } \mathcal{L}(N) \textit{ then } M_1 \xrightarrow{\alpha} M_2 \textit{ in } \mathcal{L}(N').$$

and given any two markings M_1' and M_2' in $\mathcal{L}(N')$ and any $\alpha \in \Lambda \cup \{\epsilon(n)|n \in \mathbb{N}\}$

$$\textit{if } M_1' \xrightarrow{\alpha} M_2' \textit{ in } \mathcal{L}(N') \textit{ then } M_1 \xrightarrow{\alpha} M_2 \textit{ in } \mathcal{L}(N).$$

□

Corollary 3.1.
$\mathcal{N}(\mathbb{N}, Interv_3) \cong \mathcal{N}(\mathbb{N}, Interv_2) \cong \mathcal{N}(\mathbb{N}, Interv_1)$.

We can depict a hierarchy of the classes of TAPNs as shown in Figure 15, where the arrows lead from a more general class to a less general one.

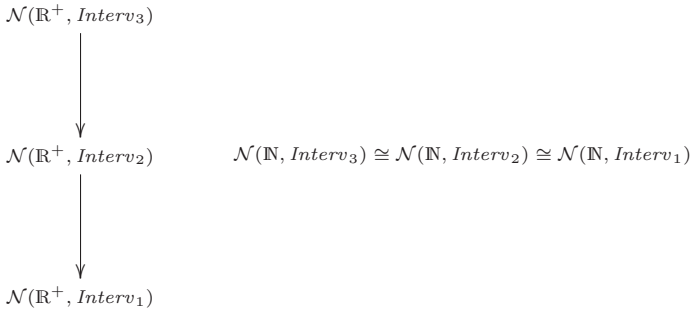


Figure 15: Class hierarchy for TAPNs

The question of whether the hierarchy of the classes with continuous time is strict, is still open.

3.2.2 TAPN as Labeled Transition System

Timed-arc Petri nets can be represented as labeled transition systems.

Definition 3.25.
*A **labeled TAPN**, in the class $\mathcal{N}(\mathcal{D}, Interv)$, is a 6-tuple $N = (P, T, In, Out, \Lambda, \lambda)$ where (P, T, In, Out) from $\mathcal{N}(\mathcal{D}, Interv)$ is a TAPN and*

- Λ *is finite set of labels*
- $\lambda : T \to \Lambda$ *is a labeling function that gives each transition $t \in T$ a label*

Definition 3.26.
*We define $\mathcal{L}(N) = (S, Act, \to)$ as the **labeled transition system** generated by a labeled TAPN $N = (P, T, In, Out, \Lambda, \lambda)$ in $\mathcal{N}(\mathcal{D}, Interv)$ s.t.*

- $S = [P \to \mathcal{D}^{\oplus}]$ *is a set of states (markings in* N*)*
- $Act = \Lambda \cup \{\epsilon(d) | d \in \mathcal{D}\}$ *is the set of labels*
- $\to \subseteq S \times Act \times S$*, is the set of allowed transitions defined by:*
  - $M \xrightarrow{a} M'$ *if and only if* $\exists t \in T$ *s.t.* $M[t\rangle M'$ *and* $a = \lambda(t)$
  - $M \xrightarrow{\epsilon(d)} M'$ *if and only if* $M[\epsilon(d)\rangle M'$*, where* $d \in \mathcal{D}$

Let us have a look at an example. Given the TAPN N with initial marking $M_0 = \{(p_1, 0), (p_4, 0), (p_4, 0), (p_5, 0)\}$ in Figure 16 we generate the LTS $\mathcal{L}(N)$, a part of which is shown in Figure 17.

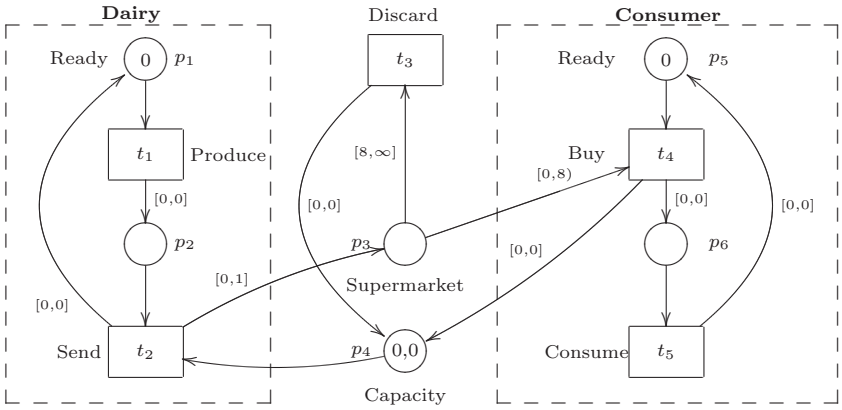


Figure 16: The TAPN N with initial marking M_0

3.2.3 Behavioral Properties

As for regular Petri nets, we can check various properties of a system modeled with a timed-arc Petri net. We will take a look at the same properties as earlier i.e. deadlock, liveness, reachability and coverability.

Definitions

In Definition 3.23 we defined a firing sequence for TAPNs, where we took into account both transition firing and the passage of time. With this new definition, the definitions of deadlock (Definition 3.14), liveness (Definition 3.15) and reachability (Definition 3.10) hold for TAPNs as well.

- A net can deadlock if in some marking no transitions are potentially fireable
- A net is live if all transitions are potentially fireable in every reachable marking

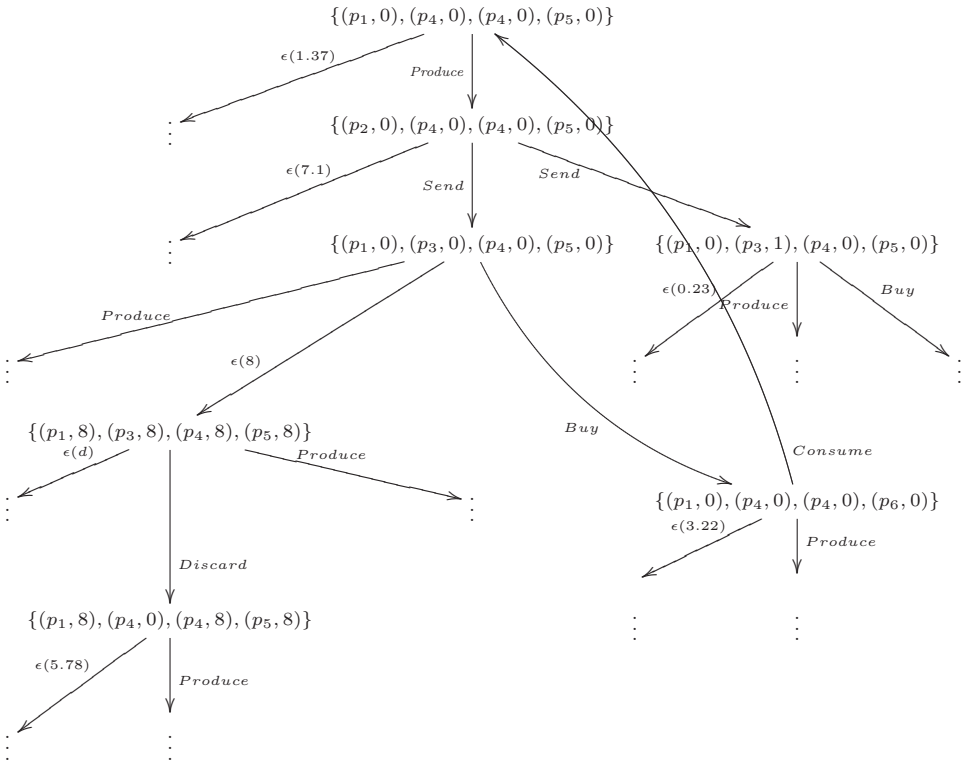


Figure 17: The LTS $\mathcal{L}(N)$

- A marking M can be reached if it is in the TAPNs reachability set.

The only definition we need to revise is coverability.

Coverability

The coverability question asks if we can reach a marking that has at least some given number of tokens, each of some minimum age, in every place.

Definition 3.27.
A marking M' on a timed-arc Petri net N ***covers*** *a marking M (written $M \subseteq M'$) if:*

$$M(p) \subseteq M'(p) \quad \forall p \in P$$

A marking M' covers a marking M if the tokens in each place in the net in M are a subset of the tokens in the place in M'.

Definition 3.28.
The ***coverability*** *problem for MTAPN (N, M_0) with a given marking M, is the question whether or not we can reach a marking M' that covers M from the initial marking, i.e. if $\exists M'$ s.t. $M' \in R(N, M_0)$ and $M \subseteq M'$.*

Following these definitions, we state the following lemma:

Lemma 3.4.
If a marking M' on a TAPN N covers a marking M on N then $R(N, M) \subseteq R(N, M')$, i.e. the reachability sets are monotone.

3.3 Summary

In this section we defined both the Petri net model and a timed extension of it, the timed-arc Petri net model. We have defined how they behave and looked at examples of their use. Behavioral properties for both model, properties which are useful to know about a system modeled with them, have been defined and their decidability for regular Petri nets has been summarized. In the Sections 4 and 5 we will look at decidability and undecidability results for these properties in the timed-arc Petri net model.

4 Undecidability Results for Timed-Arc Petri Nets

4.1 Undecidability of Deadlock

In [6] a polynomial time reduction from reachability to deadlock is shown for regular Petri nets. We use this approach as an inspiration when creating a reduction for TAPNs.

Let us fix $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ and $Interv \in \{Interv_1, Interv_2, Interv_3\}$.

Theorem 4.1.
Reachability is polynomial-time reducible to deadlock.

Proof.
Given a TAPN $N = (P, T, In, Out)$, in any class $\mathcal{N}(\mathcal{D}, Interv)$, a marking M on it and the initial marking M_0, we construct a new TAPN $N' = (P', T', In', Out')$ in the same class, with the initial marking M'_0 s.t. if M can be reached in N then N' will always deadlock.

We assume for now that the ages of the tokens in the final marking M are natural numbers, but this is not really a restriction like we show later, were stretching of intervals will be introduced. Stretching will allow the ages of tokens in M to be rational numbers. Real numbers will not be taken into consideration since such numbers can not be used as an input into computer programs.

Note that P will sometimes be called the original places and similarly, T will sometimes be called the original transitions.

$P' = P \cup \{run\} \cup \{kill\} \cup \{verify\} \cup \{c_p | p \in P\}$
$T' = T \cup \{t_p, loop_p, sub_p | p \in P\} \cup \{terminate\} \cup \{loop_{run}\} \cup \{loop_{kill}\} \cup \{startverify\}$

$In'(p,t) = In(p,t)\ \forall t \in T.\ \forall p \in P$
$Out'(t,p) = Out(t,p)\ \forall t \in T.\ \forall p \in P$
$In'(run,t) = \{[0,\infty]\}\ \forall t \in T$
$Out'(t,run) = \{[0,\infty]\}\ \forall t \in T$

$In'(run, loop_{run}) = \{[0,\infty]\}$
$Out'(loop_{run}, run) = \{[0,\infty]\}$

$In'(run, terminate) = \{[0,\infty]\}$
$Out'(terminate, kill) = \{[0,0]\}$

$In'(kill, loop_{kill}) = \{[0,\infty]\}$
$Out'(loop_{kill}, kill) = \{[1,1]\}$

$In'(p, sub_p) = \{[x,x] | x \in M(p)\}\ \forall p \in P$
$In'(kill, sub_p) = \{[0,0]\}\ \forall p \in P$
$In'(c_p, sub_p) = \{[0,\infty]\}\ \forall p \in P$

$Out'(sub_p, kill) = \{[0,0]\}\ \forall p \in P$

To clarify the above, it says that for each token in a place p in the target marking M, of the original net N, one arc is created from that place to the corresponding sub_p transition in N'. Each of the arcs created has a time interval with the upper and lower limits both set to the same number, the age of the corresponding token in M.

$In'(kill, startverify) = \{[0,0]\}$
$Out'(startverify, verify) = \{[0,\infty]\}$

$In'(c_p, loop_p) = \{[0,\infty]\}\ \forall p \in P$
$In'(verify, loop_p) = \{[0,\infty]\}\ \forall p \in P$
$Out'(loop_p, c_p) = \{[0,\infty]\}\ \forall p \in P$
$Out'(loop_p, verify) = \{[0,\infty]\}\ \forall p \in P$

$In'(p, t_p) = \{[0,\infty]\}\ \forall p \in P$
$In'(verify, t_p) = \{[0,\infty]\}\ \forall p \in P$
$Out'(t_p, p) = \{[0,\infty]\}\ \forall p \in P$
$Out'(t_p, verify) = \{[0,\infty]\}\ \forall p \in P$

$$M'_0(p) = \begin{cases} M_0(p) & \text{if } p \in P \\ \{0\} & \text{if } p = run \\ \{0\} & \text{if } p = c_{p'} \text{ and } |M(p')| > 0 \\ \emptyset & \text{otherwise} \end{cases}$$

We will argue that M is reachable in N if and only if N' deadlocks. The first thing to notice is that the *terminate* transition which disables all the t transitions from the original net N, is always enabled as long as it does not fire. After it has fired it can never be enabled again. Therefore as long as it has not fired, no reachable marking is dead. Two cases for M need to be checked.

- Suppose that M is reachable in N, then we will show that N' deadlocks: $M \cup \{(run, 0)\}$ is also reachable in N' by firing exactly the same sequence of t transitions that were fired in N in order to reach M. When M has been reached in N' the *terminate* transition can be fired, creating a fresh token in the *kill* place. As there is a token in the *kill* place, all the tokens can be removed from the original places in N and from the c_p places using the sub_p transitions. Note that the sub_p has to fire as soon as the token is created in the *kill* place since the interval on the arc from *kill* to the sub_p transitions is $[0,0]$. The age of the tokens in the c_p places does not matter as the interval on the arc from those places to sub_p is $[0,\infty]$. Since we have reached M, each token in the p places should fit an interval on one arc from its place to a corresponding sub_p transition, since the intervals on these arcs were created according to the marking we wanted to reach

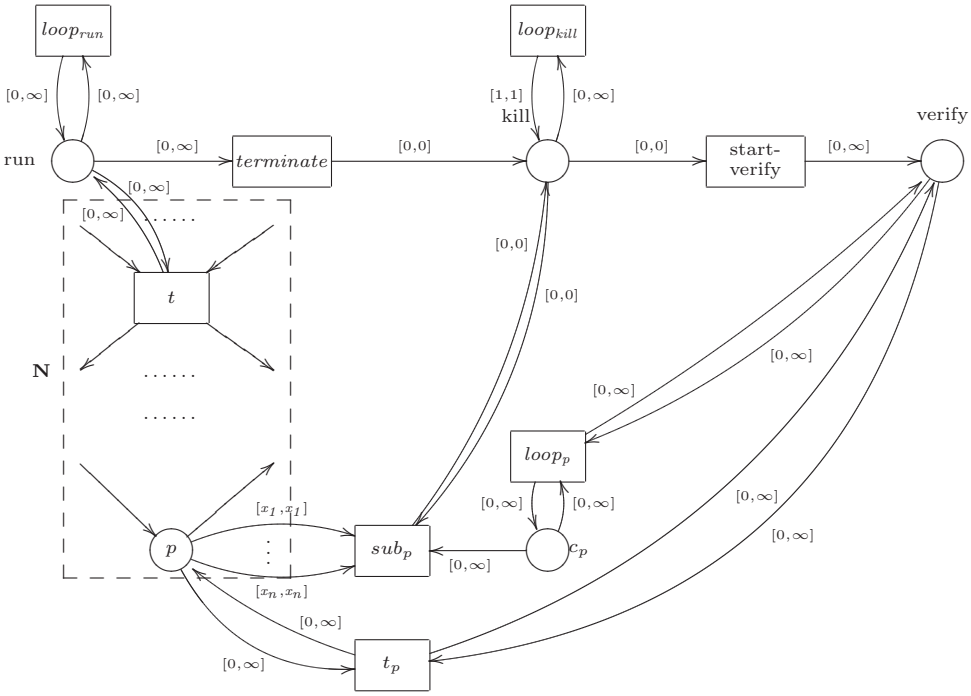


Figure 18: Reduction from reachability to deadlock

in the beginning.
Now it is possible to create a token in the *verify* place, a place that is used to check if N' has a deadlock. As all the tokens were removed from the places in the original net N, no t_p transition can fire and because all the tokens were removed from all the c_p places, no $loop_p$ can fire either. Hence N' deadlocks.

- Suppose that M is not reachable in N, then we will show that N' does not deadlock:

  Before the *terminate* transition has fired there is no deadlock because the terminate transition is always enabled (there is a token in the run place from the beginning). After *terminate* has fired all the transitions in the original net N become disabled and a new token of age 0 is created in the *kill* place.
  If the target marking M is the empty marking, it is not reachable if there are some tokens in N. The fact that M is the empty marking means that no tokens are placed in the c_p places, which again means that the sub_p transitions will never become enabled. If on the other hand M is not the empty marking some tokens will be placed in the c_p places.
  After the *terminate* transition has fired, a fresh token is created in the *kill* place. Then the option to let time elapse is always available, no matter how

M looks like, and if that option is taken the $loop_{kill}$ transition becomes the only transition that can fire. After the firing of that transition it will always remain enabled and N' does not deadlock.
Instead of letting time elapse in the *kill* place, we can also try to fire a transition. The enabled transitions depend on whether M was the empty marking or not. If M was not the empty marking then the sub_p transitions can fire and remove tokens from N and from the c_p places. If on the other hand M was the empty marking the sub_p transitions will never become enabled and therefore the only option is to fire the *startverify* transition and create a token in the *verify* place, which can also be done after firing the sub_p transitions.
In the verify phase there are some options to fire transitions. If some tokens remain in the original places of N, which is the case if M was the empty marking or if the firing of the sub_p transitions did not remove all of the tokens from N, some t_p transition will always remain enabled, no matter how time elapses, since the intervals on the arcs leading to the t_p transitions are $[0, \infty]$. If some tokens remain in the c_p places some $loop_p$ transition will always remain enabled for the same reasons as the t_p transitions remained enabled.

□

Figure 19 shows an example of a reduction from reachability to deadlock for a simple TAPN N. The initial marking is $M_0 = \{(p_1, 0), (p_1, 0), (p_1, 0)\}$ and the target marking, the marking we want to reach, is $M = \{(p_1, 1), (p_2, 1), (p_2, 2)\}$.

Stretching

As mentioned before, the assumption that the ages of the tokens in the target marking M need to be natural numbers, is not really a restriction. Let us now assume that the ages of the tokens in M can be rational numbers. We have shown that the reduction from reachability to deadlock works for natural numbers, and to show that it also works for rational numbers, *stretching* will be introduced.

Definition 4.1.
*Given a TAPN $N = (P, T, In, Out)$, from any class $\mathcal{N}(\mathcal{D}, Interv)$ and a marking M on it, where the ages of the tokens in M are rational numbers $\{x_1/y_1, x_2/y_2, \ldots, x_n/y_n\}$ ($x_i, y_i \in \mathbb{N}$ for $1 \leq i \leq n$) and N has intervals of the form $[\![a, b]\!]$ ($a, b \in \mathbb{N}$). The intervals can be **stretched** by multiplying the upper and lower limits, a and b respectively, with the least common denominator of the ages of tokens $\{x_1/y_1, x_2/y_2, ..., x_n/y_n\}$. Each token age is then multiplied with the same number, making it a natural number.*

This means that the ages of the tokens can be changed to natural numbers and the intervals stretched accordingly, preserving, amongst others, the reachability property of the underlying net. This is a fact since the least common

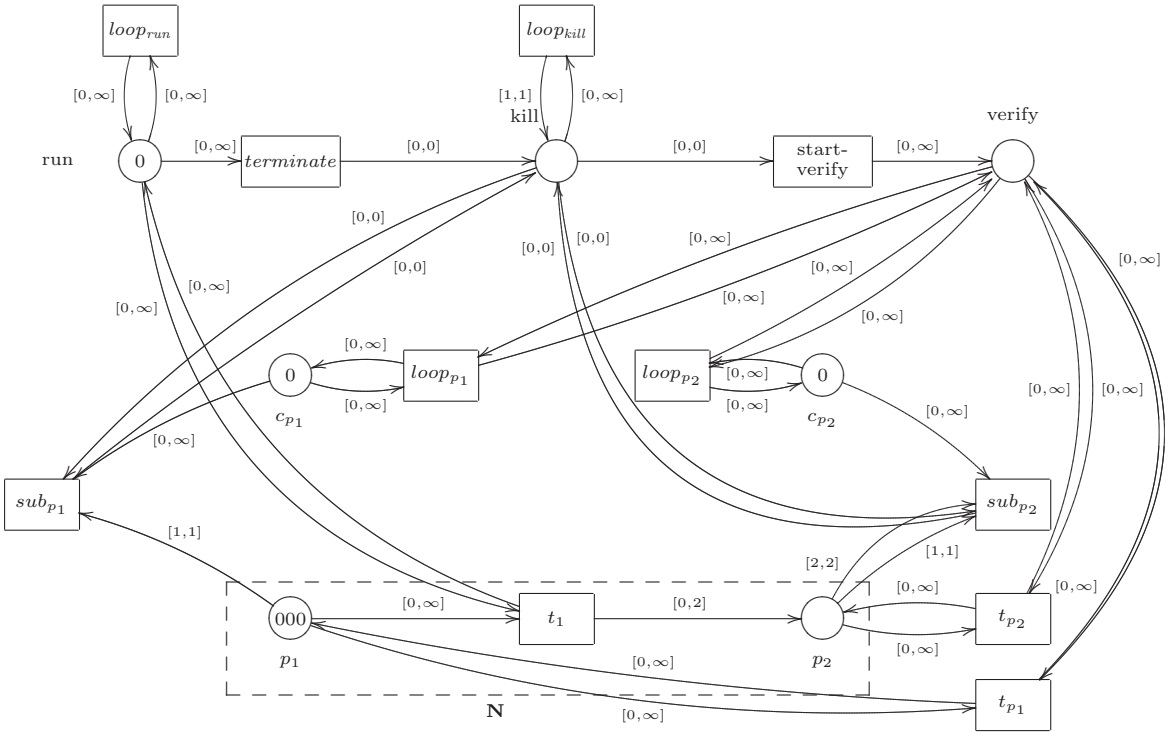


Figure 19: Reachability to deadlock example

denominator of the token ages is always a natural number, so are the upper and lower limits of each of the intervals and when two natural numbers are multiplied together, the outcome is a natural number. Therefore the restriction on the intervals that they need to be natural numbers, holds, and each token age becomes a natural number as well.

Corollary 4.1.
Deadlock is undecidable for TAPNs from any class $\mathcal{N}(\mathcal{D}, Interv)$.

Proof.
In [18] reachability is shown to be undecidable for TAPNs. We have shown that reachability is polynomial time reducible to deadlock and thus that deadlock is also undecidable for TAPNs. □

4.2 Undecidability of Liveness

In [6] a polynomial time reduction from deadlock to liveness is shown for regular Petri nets. We use this approach as an inspiration when creating a reduction for TAPNs.

Let us fix $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ and $Interv \in \{Interv_1, Interv_2, Interv_3\}$.

Theorem 4.2.
Deadlock is polynomial-time reducible to liveness

Proof.
Given a TAPN $N = (P, T, In, Out)$, in any class $\mathcal{N}(\mathcal{D}, Interv)$, with initial marking M_0 we construct a new TAPN $N' = (P', T', In', Out')$, in the same class, with initial marking M'_0, s.t. if N deadlocks then N' is always live, as follows:

$P' = P \cup \{ok\}$
$T' = T \cup \{t' | t \in T\} \cup \{live\}$

$In'(p,t) = In(p,t)\ \forall t \in T. \forall p \in P$
$Out'(t,p) = Out(t,p)\ \forall t \in T. \forall p \in P$

$In'(p,t') = In(p,t)\ \forall t \in T. \forall p \in P$
$Out'(t',ok) = \{[0,0]\}\ \forall t' \in T'$

$In'(ok, live) = \{[0,\infty]\}$
$Out'(live,p) = \{[0,\infty]\}\ \forall p \in P$
$Out'(live,ok) = \{[0,0]\}$

$$M'_0(p) = \begin{cases} M_0(p) & \text{if } p \in P \\ \emptyset & \text{if } p = ok \end{cases}$$

N has no reachable dead marking if and only if N' is live. To show this we will look at two cases:

- If N can reach a dead marking M_{dead}, then N' is not live:
  If N can reach M_{dead} then N' can reach the same marking without firing any t' transition. Since the t' transitions in N' have identical input arcs to the t transitions in N, M_{dead} is a dead marking in N' and therefore N' is not live.

- If N has no reachable dead marking then N' is live:
  This means that any reachable marking is not dead so we can fire at least one t' transition. Doing this creates a token in the *ok* place, which can then never be emptied. That means that the *live* transition can always fire, and since it creates tokens in all places in N', every transition in N' is live, making N' live.

□

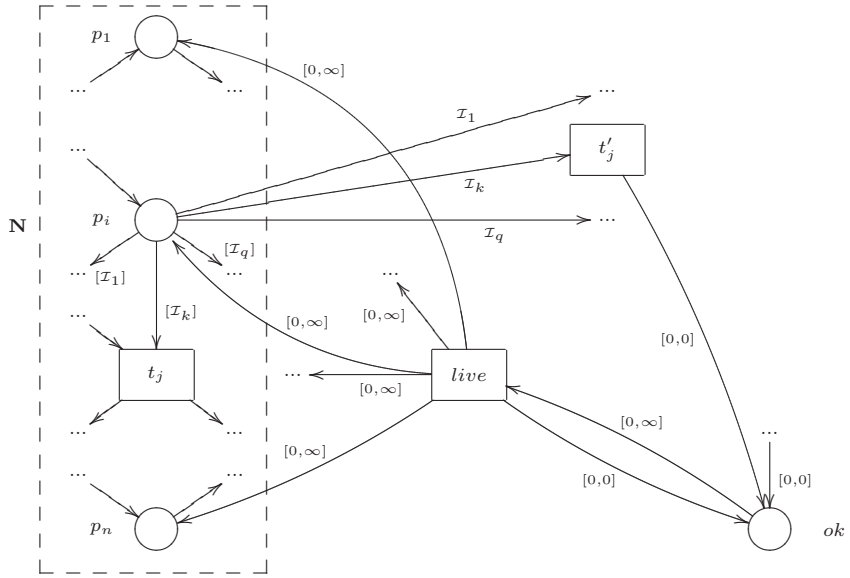


Figure 20: Reduction from deadlock to liveness

Figure 21 shows an example of a reduction from deadlock to liveness for a simple TAPN N.

Corollary 4.2.
Liveness is undecidable for TAPNs from any class $\mathcal{N}(\mathcal{D}, Interv)$.

Proof.
In the previous section we proved that deadlock is undecidable for TAPNs. Now we have shown that deadlock is polynomial time reducible to liveness and thus that liveness is also undecidable for TAPNs. □

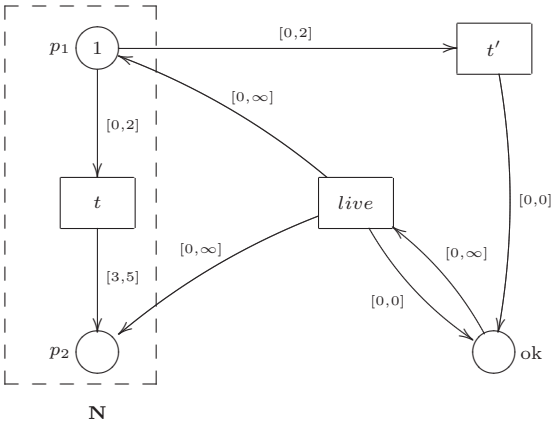


Figure 21: Deadlock to liveness example

4.3 Summary

Polynomial time reductions for regular Petri nets have been a focus of research for years. A summary of the relations of reachability, deadlock and liveness, with regard to polynomial time reductions, are shown in Figure 22.

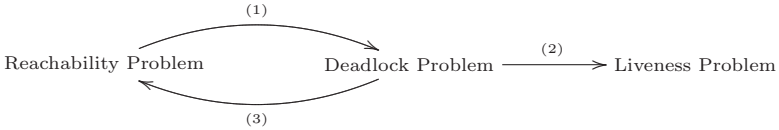


Figure 22: Summary of polynomial time reductions for PNs

(1) In [6] the authors give a polynomial time reduction from reachability to deadlock and (2) a polynomial time reduction from deadlock to liveness as well. (3) In [9] reachability and deadlock are shown to be polynomially equivalent. In [12] the author shows that liveness and reachability are recursively equivalent, but a polynomial time reduction between them remains an open problem.

For TAPNs, with inspiration from [6], we have constructed polynomial time reductions from reachability to deadlock and from deadlock to liveness. Since we know that reachability is undecidable for TAPNs [18] we have been able to proof that both deadlock and liveness are undecidable for TAPNs as well. Figure 23 shows a summary of the relations between these three properties, with regard to polynomial time reductions.

Figure 23: Summary of polynomial time reductions for TAPNs

In [20] the author shows that reachability for 1-safe TAPNs is PSPACE-Complete. With that knowledge and our reductions we can see that both deadlock and liveness are PSPACE-Hard. To show that they are PSPACE-Complete we need to show that they are polynomially equivalent to reachability but for now, that remains an open problem.

Claim 4.1.
We claim that deadlock for 1-safe TAPNs is PSPACE-Complete.

It has been shown for regular Petri nets that reachability and deadlock are polynomially equivalent. Reachability for 1-safe TAPNs was further more proved to be PSPACE-Complete in [20] and since we have found a polynomial time reduction from reachability to deadlock we claim that there exists a polynomial time reduction from deadlock to reachability for 1-safe TAPNs making those two properties polynomially equivalent and thus making deadlock PSPACE-Complete.

5 Deciding Coverability for Timed-Arc Petri Nets

In [3] the authors show how to use backwards reachability analysis and existential zones to decide coverability for timed-arc Petri Nets.

5.1 Backwards Reachability Analysis

The backwards reachability analysis [17] can be applied to any labeled transition system, to check if a given state is reachable from another.

The key idea is to check whether a given state s' is reachable from an initial state s by starting at s' and trace backwards the possible transitions that could be performed to reach this state. By doing this iteratively, and examining if at some point we discover the initial state s, we can determine if the state s' is reachable from s, i.e. $s \xrightarrow{*} s'$.

More to our interest, the backwards reachability analysis can also be used to determine the reachability of a set of states $K \subseteq S$ from a given initial state s_0 in a transition system $\mathcal{L} = (S, Act, \rightarrow)$. Let $pre_\alpha(K)$ be the set of states from which we can do the single transition $\alpha \in Act$ to reach a state in K. Formally,

$$pre_\alpha(K) = \{s \in S | \exists s' \in K \text{ s.t. } s \xrightarrow{\alpha} s'\} \text{ (Figure 24).}$$

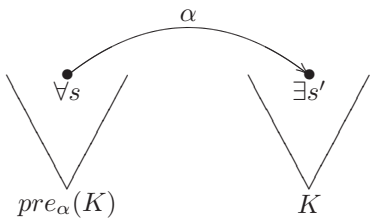


Figure 24: The sets $pre_\alpha(K)$ and K

As we iteratively trace all possible transitions in Act backwards, we generate a $pre_\alpha(K)$ set for every $\alpha \in Act$. Let

$$pre(K) = \bigcup_{\alpha \in Act} pre_\alpha(K)$$

denote the set of all states from which we can do a transition to reach a state in K, i.e.

$$\forall s \in pre(K)\ \exists s' \in K\ s.t.\ s \xrightarrow{\alpha} s'\ for\ some\ \alpha \in Act.$$

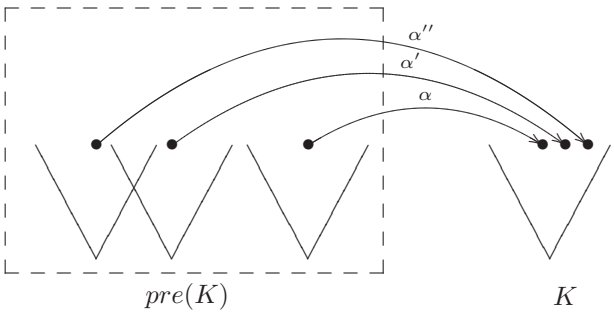


Figure 25: The sets $pre(K)$ and K

Then $pre^*(K)$ denotes the transitive closure

$$K_0 = K$$
$$K_1 = pre(K_0)$$
$$K_2 = pre(K_1)$$
$$\vdots$$
$$K_{i+1} = pre(K_i)$$

Hence, $pre^*(K) = \bigcup_{i \geq 0} K_i$ is the set of all states from which we can do zero or more transitions to reach a state in K (Figure 25). Checking if $s_0 \in pre^*(K)$ is then a simple task, given that $pre^*(K)$ is finite.

5.2 Existential Zones

Notice that when dealing with infinite state systems like TAPNs, the set $pre^*(K)$ might be an infinite set. It is obvious, that if it is infinite, we cannot check if $s \in pre^*(K)$. So, for this technique to be applied to infinite-state systems, we need to represent $pre^*(K)$ in a finite way.

The theory of *existential zones (EZ)* is based on the well known theory of *existential regions (ER)* which has been applied to verify timed networks [4]. These constraint systems are variants of other constraint systems, zones and regions respectively. The difference is that EZ and ER operate on unbounded number of clocks, opposite to zones and regions, which operate on finite sets of clocks and are therefore not directly suitable for analyzing timed-arc Petri nets.

The intuitive notion of existential zones, is that they provide a way by which we can represent an infinite set of markings in a finite way.

The theory of EZ was first proposed in [4] as an analysis technique to be used in backwards reachability analysis of timed-arc Petri nets. Existential zones characterize a set of markings and enable us to represent an infinite set of markings

in a finite way. EZ as represented in [4] can be applied to all TAPNs that belong to the class $\mathcal{N}(\mathbb{R}^+, Interv_1)$. We extend this technique to handle all TAPNs belonging to the class $\mathcal{N}(\mathbb{R}^+, Interv_3)$.

Notation:
We use the notation $\underline{n}^0$ to denote the set $\{0, 1, \ldots, n\}$ and $\underline{n}^1$ to denote the set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$.

Intuition:
A **difference bound matrix (DBM)** is a matrix of size $(m+1) \times (m+1)$, where $m \in \mathbb{N}$, used to represent constraints on ages of m artificial tokens.

$D : \underline{m}^0 \times \underline{m}^0 \to \mathbb{Z} \cup \{\infty\} \cup \overline{\mathbb{Z}}$ where $\overline{\mathbb{Z}} = \{\overline{z} | z \in \mathbb{Z}\}$.

For a DBM D, we write $D(k, l)$ to denote the value in row k and column l. A DBM specifies minimum constraints on the ages of tokens x_i and x_j, for each $i, j \in \underline{m}^1$, s.t.:

- $\begin{cases} x_i \leq D(i,0) & \text{if } D(i,0) \in \mathbb{Z} \\ x_i < D(i,0) & \text{if } D(i,0) \in \overline{\mathbb{Z}} \end{cases}$
- $\begin{cases} -D(0,i) \leq x_i & \text{if } D(0,i) \in \mathbb{Z} \\ -D(0,i) < x_i & \text{if } D(0,i) \in \overline{\mathbb{Z}} \end{cases}$
- $\begin{cases} x_i - x_j \leq D(i,j) & \text{if } D(i,j) \in \mathbb{Z} \\ x_i - x_j < D(i,j) & \text{if } D(i,j) \in \overline{\mathbb{Z}} \end{cases}$

The overlined values in the DBM indicate that the constraints are with strict inequalities. Note that the diagonal line in the DBM is irrelevant.

Let us look at an example where we want to represent the following constraints on some two tokens, artificially represented by x_1 and x_2:

- $x_1 \leq 10$
- $x_2 > 6$
- $x_2 - x_1 \leq 2$

With these constraints we construct the following DBM D:

$$D = \begin{array}{c|ccc} {}_i\backslash^j & 0 & 1 & 2 \\ \hline 0 & - & 0 & \overline{-6} \\ 1 & 10 & - & \infty \\ 2 & \infty & 2 & - \end{array}$$

D represents the constraints given above in the following manner:

- $D(1,0) = 10 : \quad x_1 \leq 10$

- $D(0,2) = \overline{-6}: \quad -(-6) < x_2$
- $D(2,1) = 2: \quad x_2 - x_1 \leq 2$

The difference bound matrix plays an essential role in existential zones.

Definition 5.1.
*An **existential zone** is a triple $Z = (m, \overline{P}, D)$ where:*

- $m \in \mathbb{N}$ *is the minimum number of tokens required*
- $\overline{P}$ *(*placing*) is a function $\overline{P} : \{x_1, \ldots, x_m\} \to P$, where $\{x_1, \ldots, x_m\}$ is a set of artificial tokens, pointing to real tokens in the marking. $\overline{P}$ specifies where tokens should be located*
- D *is a difference bound matrix of the size $(m+1) \times (m+1)$*

An existential zone states the minimum requirements on a marking for it to be *satisfactory*:

- The minimum number of tokens the marking should have
- The placing specifies where the tokens should be located
- The DBM specifies minimum constraints on the ages of tokens and the difference between the ages of any two tokens

We say that a marking M *satisfies* an existential zone Z, written $M \models Z$, according to the following definition.

Definition 5.2.
*A marking $M = \{(p_1, x_1), \ldots, (p_n, x_n)\}$ of a TAPN $N = (P, T, In, Out)$ is said to **satisfy an existential zone** $Z = (m, \overline{P}, D)$ (written $M \models Z$) if there exists an injective function $h : \underline{m}^1 \to \underline{n}^1$ (called a **witness**) s.t. the following holds:*

- *(1)* $\overline{P}(x_i) = p_{h(i)}$, *for each* $i \in \underline{m}^1$
- *(2)* $\begin{cases} x_{h(i)} \leq D(i,0) & \text{if } D(i,0) \in \mathbb{Z} \\ x_{h(i)} < D(i,0) & \text{if } D(i,0) \in \overline{\mathbb{Z}} \end{cases}$ *for each* $i \in \underline{m}^1$
- *(3)* $\begin{cases} -D(0,i) \leq x_{h(i)} & \text{if } D(0,i) \in \mathbb{Z} \\ -D(0,i) < x_{h(i)} & \text{if } D(0,i) \in \overline{\mathbb{Z}} \end{cases}$ *for each* $i \in \underline{m}^1$
- *(4)* $\begin{cases} x_{h(i)} - x_{h(j)} \leq D(i,j) & \text{if } D(i,j) \in \mathbb{Z} \\ x_{h(i)} - x_{h(j)} < D(i,j) & \text{if } D(i,j) \in \overline{\mathbb{Z}} \end{cases}$ *for each* $i, j \in \underline{m}^1$

Intuitively, this states:

(1) Tokens in M are located according to the placing $\overline{P}$.

(2) The ages of tokens do not violate the upper bounds represented in D.

(3) The ages of tokens do not violate the lower bounds represented in D.

(4) The difference between the ages of two tokens does not violate the constraints represented in D.

Checking if $M \models Z$ for a given marking M, is straightforward. We check if we can find a match for the actual tokens in $M = \{(p_1, x_1), \ldots, (p_n, x_n)\}$ from the artificial tokens represented in Z. This is done by finding and applying an injective function $h : \underline{m}^1 \to \underline{n}^1$ called a *witness* function, where $x_{h(i)}$ is an artificial token mapped to an actual token in M.

Example 5.1.
We define an existential zone Z_0 for the TAPN N with the marking $M = \{(p_1, 0), (p_3, 0), (p_3, 1)\}$ in Figure 26.

$Z_0 = (3, \overline{P}_0, D_0)$ *where:*

$$\begin{aligned} \overline{P}_0(x_1) &= p_1 \\ \overline{P}_0(x_2) &= p_3 \\ \overline{P}_0(x_3) &= p_3 \end{aligned}$$

and

$$D_0 = \begin{array}{c|cccc} {}_i\backslash^j & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 0 & - & \infty & \infty \\ 2 & 0 & \infty & - & -1 \\ 3 & 1 & \infty & \infty & - \end{array}$$

The zone Z_0 thus requires at least 3 tokens, one in place p_1 and two in p_3, and the constraints on the ages of the tokens represented in the DBM say that one of the tokens in p_3 (x_3) must at least be of age 1 and that it must also be one time unit older that the other token in p_3 (x_2).

An example of a marking that satisfies this zone Z_0 is

$$M_1 = \{(p_1, 0), (p_1, 2), (p_3, 1), (p_3, 1), (p_3, 5)\},$$

as we can find in M_1 at least 3 tokens (witnesses) that match the criteria given by Z_0.

The set of all markings that satisfy Z is denoted by $[\![Z]\!] = \{M |\ M \models Z\}$. An existential zone is *consistent* if $[\![Z]\!] \neq \emptyset$. The set $[\![Z]\!]$ is upwards closed and is always infinite, for a consistent zone Z.

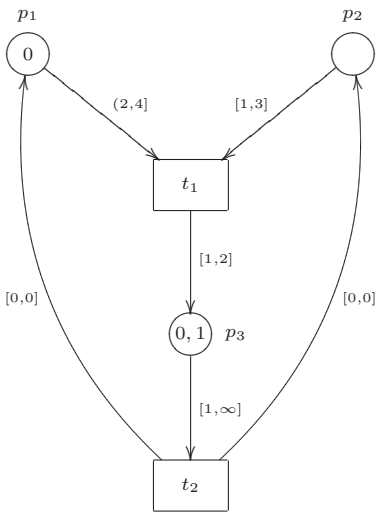


Figure 26: The TAPN N

We show that the set $[\![Z]\!]$ is always infinite for a consistent Z. If we consider the existential zone Z, we can find infinitely many markings that satisfy it. By replicating a marking M that satisfies Z and adding one or more tokens to it we create a new marking M' s.t. $M \subseteq M'$. This can be done repeatedly, i.e. we can find a marking M'' s.t. $M' \subseteq M''$, and since there are no upper bounds on the number of tokens a marking can have, we can create infinitely many markings that cover M and satisfy Z. Thus, if $[\![Z]\!]$ is consistent, it is always infinite.

Existential zones are normalized [4] if irrelevant constraints have been removed. An example are the following constraints, one on a token x_1 and two other constraints that combined imply a new constraint on x_1:

(1) $2 \leq x_1 \leq 10$
(2) $5 \leq x_2$
(3) $x_2 - x_1 \leq 2$

Constraints (2) and (3) together imply that $x_1 \geq 3$. Therefore, by normalizing these constraints in a DBM we would modify constraint (1) to be $3 \leq x_1 \leq 10$.

Definition 5.3.
An existential zone $Z = (m, \overline{P}, D)$ is said to be ***normalized*** *if for each $i, j, k \in \underline{m}^0$, we have $D(i,j) \leq D(i,k) + D(k,j)$.*

By normalizing, we combine constraints to create new ones that represent the most strict bounds on each token. A normalized existential zone is denoted by $\tilde{Z}$ and $[\![\tilde{Z}]\!] = [\![Z]\!]$[4]. Normalization enables us to compare two zones and see

if they are equal. Given two zones $Z = (m, \overline{P}, D)$ and $Z' = (m', \overline{P'}, D')$, we can normalize both of them and then check if there exists a bijection $b : m \to m'$ s.t. $b(\overline{P}) = \overline{P'}$ and $b(D) = D'$ where:

$$b(\overline{P})(x_i) = \overline{P}(x_{b(i)})$$

$$b(D)(i,j) = D(b(i), b(j))$$

From now on we will only consider normalized zones.

5.3 Applying Backwards Reachability Analysis with Existential Zones

We apply the backwards reachability analysis using existential zones to decide coverability for timed-arc Petri nets.

Theorem 5.1.
Coverability is decidable for timed-arc Petri nets in the class $\mathcal{N}(\mathbb{R}^+, Interv_3)$.

The proof of Theorem 5.1 was done in [8] for TAPNs in $\mathcal{N}(\mathbb{R}^+, Interv_1)$. The proof provided in this section applies to TAPNs in $\mathcal{N}(\mathbb{R}^+, Interv_3)$. Recall the coverability problem, where we want to know if we can reach any marking M' that covers a given marking M from the initial marking M_0. Given a TAPN $N = (P, T, In, Out)$ in $\mathcal{N}(\mathbb{R}^+, Interv_3)$.

We start by defining an existential zone Z_0 that defines the minimum constraints of a marking according to M. We know that any other marking M' that satisfies Z_0 is a marking that covers M, i.e. $M' \models Z_0 \Rightarrow M \subseteq M'$.

When Z_0 has been defined, we start applying the backwards reachability analysis described earlier in this section. We trace back each transition $t \in T$ and discover each $pre_t(Z)$ zone and we trace back the time-elapsing transition $\{\epsilon(r) | r \in \mathbb{R}^+\}$ to discover the $pre_\epsilon(Z)$ zone as well. Then we have discovered a set of zones

$$pre(Z) = \{pre_\epsilon\} \cup \bigcup_{t \in T} \{pre_t(Z)\}$$

which is always finite, since the number of these transitions is always finite (later we will show $pre_\epsilon(Z)$ is computed as only one zone for all time delays $r \in \mathbb{R}^+$).

After we have calculated $pre(Z)$ we have a set of existential zones that characterizes all markings from which we can reach a marking in Z_0 in one step.

We repeat this procedure for all the existential zones in $pre(Z)$ to discover the next set $pre(pre(Z)) = \bigcup_{Z' \in pre(Z)} pre(Z')$. We do this iteratively until we have calculated the whole transitive closure $pre^*(Z)$ where we reach a fixed point s.t. $pre(pre'(Z)) = pre'(Z)$. This termination of the iteration is guaranteed since EZ are better-quasi ordered [3].

When we know $pre^*(Z)$, we can check if the initial marking M_0 satisfies any of the existential zones in $pre^*(Z)$ (see how in Defintion 5.2) and hence, we can

determine if we can reach a marking M' that covers a given marking M from an initial marking M_0.

To complete this proof we show how the zones $pre_t(Z)$ and $pre_\epsilon(Z)$ are computed.

Computing $pre_t(Z)$
To calculate $pre_t(Z)$ we define the cases for comparison of values in $\mathbb{Z}$ and $\overline{\mathbb{Z}}$. We first recall the definition of $min(a, b)$ where $a, b \in \mathbb{Z}$:

$$min(a,b) = \begin{cases} \text{a} & \text{if } a \leq b \\ \text{b} & \text{if } a > b \end{cases}$$

Now we can define the min function for values in $\mathbb{Z} \cup \overline{\mathbb{Z}}$.

$$min(x,y) = \begin{cases} a & \text{if } a \leq b \text{ and } x = a \in \mathbb{Z}, y = b \in \mathbb{Z} \\ b & \text{if } a > b \text{ and } x = a \in \mathbb{Z}, y = b \in \mathbb{Z} \\ \\ \overline{a} & \text{if } a \leq b \text{ and } x = \overline{a} \in \overline{\mathbb{Z}}, y = \overline{b} \in \overline{\mathbb{Z}} \\ \overline{b} & \text{if } a > b \text{ and } x = \overline{a} \in \overline{\mathbb{Z}}, y = \overline{b} \in \overline{\mathbb{Z}} \\ \\ a & \text{if } a < b \text{ and } x = a \in \mathbb{Z}, y = \overline{b} \in \overline{\mathbb{Z}} \\ \overline{b} & \text{if } a \geq b \text{ and } x = a \in \mathbb{Z}, y = \overline{b} \in \overline{\mathbb{Z}} \\ \\ \overline{a} & \text{if } a \leq b \text{ and } x = \overline{a} \in \overline{\mathbb{Z}}, y = b \in \mathbb{Z} \\ b & \text{if } a > b \text{ and } x = \overline{a} \in \overline{\mathbb{Z}}, y = b \in \mathbb{Z} \end{cases}$$

For example, $min(\overline{2}, \overline{3}) = \overline{2}$, $min(2, \overline{2}) = \overline{2}$ and $min(\overline{2}, 3) = \overline{2}$.

Then we define three operations, *conjunction*, *addition* and *restriction* [3].

Conjunction: The *conjunction* operation (denoted by $\otimes$) adds an additional constraint on the age of an existing token in Z.

Definition 5.4.
For an interval $\mathcal{I} = [\![a, b]\!]$, *an existential zone* $Z = (m, \overline{P}, D)$ *and* $i \in \underline{m^1}$, ***conjunction*** $Z \otimes (\mathcal{I}, i)$ *is the existential zone* $Z' = (m, \overline{P}, D')$ *where:*

- $\begin{cases} D'(i,0) = min(b, D(i,0)) & \text{if } \mathcal{I} = [\![a, b]\!] \\ D'(i,0) = min(\overline{b}, D(i,0)) & \text{if } \mathcal{I} = [\![a, b)\!] \end{cases}$

- $\begin{cases} D'(0,i) = min(-a, D(0,i)) & \text{if } \mathcal{I} = [a,b] \\ D'(0,i) = min(-\overline{a}, D(0,i)) & \text{if } \mathcal{I} = (a,b] \end{cases}$
- $D'(k,j) = D(k,j)$ *for each* $j,k \in \underline{m}^1$ *with* $j \neq k, (k,j) \neq (i,0)$ *and* $(k,j) \neq (0,i)$

Intuitively, to create Z', we modify the first row and the first column so that the DBM D' in Z' represent the tightest constraint that can be composed from the constraints in D and the constraints we add. For example, for a zone

$$Z = \left(2, \overline{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array} \right)$$

we perform the operation $Z \otimes ([1,10],1)$ to get the following Z':

$$Z' = \left(2, \overline{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & -1 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array} \right)$$

Addition: The second operation, *addition* (denoted by $\oplus$) simply adds a new token to a zone Z.

Definition 5.5.
For an existential zone $Z = (m, \overline{P}, D)$*, a place* $p \in P$ *and an interval* $\mathcal{I} = [\![a,b]\!]$*,* ***addition*** $Z \oplus (p, \mathcal{I})$ *gives us an existential zone* $Z' = (m+1, \overline{P'}, D')$ *where*

- $\overline{P'}(m+1) = p$*, and* $\overline{P'}(j) = \overline{P}(j)$ *for each* $j \in \underline{m}^1$
- $\begin{cases} D'(0,m+1) = a & \text{if } \mathcal{I} = [a,b] \\ D'(0,m+1) = \overline{a} & \text{if } \mathcal{I} = (a,b] \end{cases}$
- $\begin{cases} D'(m+1,0) = b & \text{if } \mathcal{I} = [\![a,b] \\ D'(m+1,0) = \overline{b} & \text{if } \mathcal{I} = [\![a,b) \end{cases}$
- $D'(m+1,j) = \infty$ *and* $D'(j,m+1) = \infty$*, for each* $j \in \underline{m}^1$
- $D'(k,j) = D(k,j)$ *for each* $j,k \in \underline{m}^0$ *with* $j \neq k$

Intuitively, we add a column and a row to the DBM D in Z to create D' in Z'. The values in the new column/row are according to the interval $\mathcal{I}$ for the 0 row and column, and other values are set to infinity. We add one token to the placing and increase m by one. For example, for a zone

$$Z = \left(2,\ \begin{matrix} \overline{P}(1) = B \\ \overline{P}(2) = C \end{matrix},\ \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array} \right)$$

we perform the operation $Z \oplus (A, [1, 2])$ to get the following Z':

$$Z' = \left(3,\ \begin{matrix} \overline{P}(1) = B \\ \overline{P}(2) = C \\ \overline{P}(3) = A \end{matrix},\ \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 8 & \infty \\ 2 & 8 & 4 & - & \infty \\ 3 & 2 & \infty & \infty & - \end{array} \right)$$

Abstraction: Finally, the *abstraction* operation removes a given token i from an existential zone Z (denoted by Z/i).

Definition 5.6.
*For an existential zone $Z = (m, \overline{P}, D)$, and a token i, **abstraction** Z/i gives us an existential zone $Z' = (m-1, \overline{P'}, D')$ where*

- $\overline{P'}(j) = \overline{P}(j)\ \forall j \in \underline{(i-1)}^0$, *and* $\overline{P'}(j) = \overline{P}(j+1)$, *for* $j \in \{i, \ldots, m-1\}$.
- $D'(j,k) = D(j,k)\ \forall j, k \in \underline{(i-1)}^0$
- $D'(j,k) = D(j,k+1)$ *and* $D'(k,j) = D(k+1,j)\ \forall j \in \underline{(i-1)}^0$ *and* $k \in \{i, ..., m-1\}$
- $D'(j,k) = D(j+1,k+1)\ \forall j, k \in \{i, ..., m-1\}$

Intuitively, we remove one row and one column from the DBM D in Z to create the D' in Z', remove one token from the placing and decrease m by one. For example, for a zone

$$Z = \left(3,\ \begin{matrix} \overline{P}(1) = B \\ \overline{P}(2) = C \\ \overline{P}(3) = A \end{matrix},\ \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 8 & \infty \\ 2 & 8 & 4 & - & \infty \\ 3 & 2 & \infty & \infty & - \end{array} \right)$$

we perform the operation $Z/2$ to get the following Z':

$$Z' = \left(2,\ \begin{matrix} \overline{P}(1) = B \\ \overline{P}(2) = A \end{matrix},\ \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & -1 \\ 1 & 8 & - & \infty \\ 2 & 2 & \infty & - \end{array} \right)$$

We are now ready to define $pre_t(Z)$ [3]:

Definition 5.7.
Consider a TAPN $N = (P, T, In, Out)$, a transition $t \in T$, and an existential zone $Z = (m, \overline{P}, D)$. Let $In(t) = \{(p_1, \mathcal{I}_1), \ldots, (p_k, \mathcal{I}_k)\}$, and $Out(t) = \{(q_1, \mathcal{J}_1), \ldots, (q_\ell, \mathcal{J}_\ell)\}$

Then $pre_t(Z)$ is the smallest set containing each existential zone Z' such that there is a partial injection $h : \underline{m}^1 \to \underline{\ell}^1$ with a domain $\{i_1, ..., i_n\}$, and an existential zone Z' satisfying the following conditions:

(1) $\overline{P}(i) = q_{h(i)}, \forall i \in \{i_1, \ldots, i_n\}$
(2) $Z \otimes (\mathcal{J}_{h(i_1)}, i_1) \otimes ... \otimes (\mathcal{J}_{h(i_n)}, i_n)$ is consistent
(3) $Z' = (Z/i_1/.../i_n) \oplus (p_1, \mathcal{I}_1) \oplus ... \oplus (p_k, \mathcal{I}_k)$

So, to calculate the zone $Z' = pre_t(Z)$ we start by mapping the artificial tokens in the placing to actual tokens in the net (1). Then we add restrictions to Z according to the intervals on the arcs from the transition t to its output places, and check if Z is consistent (2). In (3) we create a new existential zone Z', that equals Z after removing the selected tokens that are in the output places of the transition t, and then add constraints to Z' according to the intervals on arcs of the input places of t to see in which places (input places of t) we require tokens and their ages. This is how we, step by step, calculate the existential zone $Z' = pre_t(Z)$.

Computing $pre_\epsilon(Z)$
Finally, we define $pre_\epsilon(Z)$ [3]:

Definition 5.8.
For an existential zone $Z = (m, \overline{P}, D)$, the set $pre_\epsilon(Z)$ is the existential zone $Z' = (m, \overline{P}, D')$ where $D'(0, i) = 0, \forall i \in \underline{m}^1$ and $D'(j, i) = D(j, i)$ $\forall i, j \in \underline{m}^0$, with $i \neq j$ and $j \neq 0$.

To calculate the existential zone for $pre_\epsilon(Z)$, we simply lower the lower bounds of all the tokens to 0, i.e. we remove all constraints on lower bounds on the ages of tokens.

Following theorems are shown in [3]:

Theorem 5.2.
For a zone Z, $pre_t(Z)$ is the largest set of zones satisfying:

$$\forall M \in [\![pre_t(Z)]\!].\ \exists M' \in [\![Z]\!]\ s.t.\ M[t\rangle M'$$

Theorem 5.3.
For a zone Z, $pre_\epsilon(Z)$ is the largest zone satisfying:

$$\forall M \in [\![pre_\epsilon(Z)]\!].\ \exists M' \in [\![Z]\!].\ \exists r \in \mathbb{R}^+\ s.t.\ M \xrightarrow{\epsilon(r)} M'$$

Now we have shown how to calculate the sets $pre_t(Z)$ and $pre_\epsilon(Z)$. Thus we can compute $pre(Z)$. Next section illustrates in details how it works in an example.

5.4 Example

We will provide an answer to the question:
Given the TAPN N in Figure 27 can we reach a marking M' that covers the marking $M = \{(p_3, 4)\}$ from the initial marking $M_0 = \{(p_1, 0), (p_2, 0)\}$ (Figure 28)?

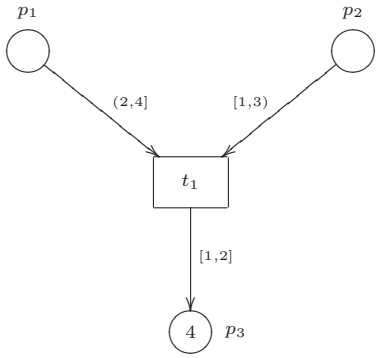


Figure 27: The TAPN N with a marking M we want to cover

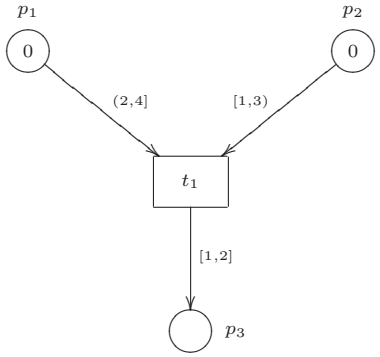


Figure 28: The initial marking M_0 of N

We start by defining the existential zone Z_0 that characterizes the marking $M = \{(p_3, 4)\}$:

$$Z_0 = \left(1,\ \overline{P}(1) = p_3\ ,\ \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & - & -4 \\ 1 & 4 & - \end{array} \right)$$

We know that any marking that satisfies Z_0 covers M.

Now we trace back each transition in the net. In this example we will only look at one case of h for each calculation, where it is an injection (not partial). Note that h can also be different than shown in the example and thus creating more zones in each iteration. Therefore, when we show the set $pre^*(Z)$, we will not show the full set, only selected zones in it.

- t_1
  For the transition t_1, with $h(1) = 1$ and $Out(t_1) = \{(q_1, [1, 2])\}$ where $q_1 = p_3$, $Z_0 \otimes ([1, 2], 1)$ is not consistent, condition (2) is violated and $pre_{t_1}(Z_0)$ does not exist.

- ϵ
  Then we calculate $pre_\epsilon(Z_0)$ to be the zone Z_1:

$$Z_1 = \left(1,\ \overline{P}(1) = p_1\ ,\ \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & - & 0 \\ 1 & 4 & - \end{array} \right)$$

Now the first iteration is complete for this h and $pre(Z_0) = \{Z_1\} \cup \{Z_0\}$.

We calculate $pre(pre(Z_0))$. Then we look at the existential zones in $pre(Z_0)$. We trace back the transition in the net to find the $pre_{t_1}(Z_1)$ and $pre_\epsilon(Z_1)$ zones.

- t_1
  For t_1, with $h(1) = 1$, $Out(t_1) = \{(q_1, [1, 2])\}$ where $q_1 = p_3$, conditions (1), (2) and (3) all hold and we get the zone $Z_2 = pre_{t_1}(Z_1)$:

$$Z_2 = \left(2,\ \begin{array}{l} \overline{P}(1) = p_1 \\ \overline{P}(2) = p_2 \end{array}\ ,\ \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & \overline{-2} & -1 \\ 1 & 4 & - & 3 \\ 2 & \overline{3} & \overline{1} & - \end{array} \right)$$

- ϵ
  Finally we find that $pre_\epsilon(Z_1) = Z_1$.

Now the second iteration is complete, for this h, and $pre(pre(Z_0)) = \{Z_2\} \cup \{Z_1\} \cup \{Z_0\}$.

We calculate $pre(pre(pre(Z_0)))$. We take the zone Z_2, and calculate $pre_{t_1}(Z_2)$ and $pre_\epsilon(Z_2)$.

- t_1
  For any h no tokens are positioned in the output place p_3 so $pre_{t_1}(Z_2)$ does not exist, since condition (1) is violated.
- ϵ

$$Z_3 = pre_\epsilon(Z_2) = \left(2, \begin{array}{l} \overline{P}(1) = p_1 \\ \overline{P}(2) = p_2 \end{array}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 4 & - & 3 \\ 2 & \overline{3} & \overline{1} & - \end{array} \right)$$

The third iteration is complete and $pre(pre(pre(Z_0))) = \{Z_3\} \cup \{Z_2\} \cup \{Z_1\} \cup \{Z_0\}$.

In the fourth iteration we discover no new zones.

- t_1
  $pre_{t_1}(Z_3)$ does not exist (violation of constraint (1)).
- ϵ
  $pre_\epsilon(Z_3) = Z_3$.

Thus, we have now completed all our iterations and discovered the transitive closure $pre^*(Z_0) = \{Z_0, Z_1, Z_2, Z_3\}$.

Let us recall our question:
Given the net N in Figure 27 can we reach a marking M' that covers the marking $M = \{(p_3, 4)\}$ from the initial marking $M_0 = \{(p_1, 0), (p_2, 0)\}$?

To determine this, we check if the given initial marking $M_0 = \{(p_1, 0), (p_2, 0)\}$ satisfies any of the zones in $pre^*(Z_0)$. We discover that $M_0 \models Z_3$, and the answer to our question is yes.

We can modify our question to ask if we can reach a marking that covers the same marking M in N from another initial marking, e.g. $M_0' = \{(p_1, 4), (p_2, 4)\}$. We would see that $M_0' \not\models Z' \; \forall Z' \in pre^*(Z)$, but since we have not computed the whole set $pre^*(Z)$ we cannot provide a complete answer but intuitively the answer to the question would be no.

5.5 Summary

The section started with an introduction of the backward reachability analysis technique. Next, we defined existential zones and how markings on TAPNs satisfy those zones. We then combined these two things and showed how backward reachability analysis can be used to decide coverability for TAPNs, with both continuous and discrete time and all types of intervals, using existential zones.

6 Extended Timed-Arc Petri Net Models

6.1 Distributed TAPN

Until now, we have discussed time dependant models where full synchronization is assumed, i.e. there is only one global clock and time elapses synchronously for the whole model. For many systems, this assumption is not justified. For example consider highly geographically distributed systems where clusters run asynchronously with one another, but the components of each cluster are synchronized.

The Distributed TAPN model was first proposed in [15] and it suggests having more than one clock in a net, making time pass synchronously only for specified parts of it.

6.1.1 The Distributed Model

Let us assume an equivalence relation $\equiv$ over places. Intuitively, $(p, p') \in \equiv$ implies that the places p and p' share a clock. When two places share a clock, the tokens placed in them grow older at the same rate. We write $p \equiv p'$ to denote that $(p, p') \in \equiv$.

Definition 6.1.
*An **equivalence class** groups together equivalent places, i.e.*

$$[p]_{\equiv} = \{p' | p \equiv p'\}$$

Let us fix $Interv \in \{Interv_1, Interv_2, Interv_3\}$ and a time domain $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$. A distributed timed-arc Petri net can then be defined as follows.

Definition 6.2.
*A **distributed timed-arc Petri net (DTAPN)** is a 5-tuple $N = (P, T, In, Out, \equiv)$ where*

- *$P = \{p_1, ..., p_n\}$ is a finite set of places*
- *$T = \{t_1, ..., t_m\}$ is a finite set of transitions s.t. $P \cap T = \emptyset$*
- *$In : P \times T \to Interv^{\oplus}$ is a function that associates to each arc (p, t) a finite multiset of intervals*
- *$Out : T \times P \to Interv^{\oplus}$ is a function that associates to each arc (t, p) a finite multiset of intervals*
- *$\equiv \subseteq P \times P$ is an equivalence relation over places*

We let $E(\equiv) = \{[p]_{\equiv} \mid p \in P\}$ denote the set of all equivalence classes in N.

Definition 6.3.
***Marked distributed timed-arc Petri net (MDTAPN)** is a pair (N, M_0), where N is a DTAPN and M_0 is an initial marking on N, s.t.*

$$\forall p \in P.\ \forall x \in M_0(p).\ x = 0$$

As an initial marking on a DTAPN N, we only allow tokens of age 0.

Definition of **firing rules** and **enabled** transitions for DTAPNs is the same as for TAPNs, Definition 6.20.

We define a time-elapsing function for MDTAPNs as follows.

Definition 6.4.
*Let (N, M_0) be a MDTAPN, where M and M' are markings on it, and $N = (P, T, In, Out, \equiv)$. We write a **time-elapsing transition** as $M[\epsilon\rangle M'$ for some time-elapsing function $\epsilon : E(\equiv) \to \mathcal{D}$ where:*

$$M'(p) = M(p) \lessdot \epsilon([p]_\equiv)\ \forall p \in P$$

Note that a TAPN is the same as a DTAPN, where all places belong to the same equivalence class, i.e. $\equiv = P \times P$.

Classes of DTAPNs
The definitions above provide a general way of defining the time attributes of the DTAPN model, both the types of intervals and the time domain. We define classes of DTAPNs as follows.

Definition 6.5.
*Let $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ be a time-domain and $Interv \in \{Interv_1, Interv_2, Interv_3\}$ be the set of allowed intervals. Then we define $\mathcal{N}_D(\mathcal{D}, Interv)$ as the **class of all DTAPNs** with time domain $\mathcal{D}$ and interval set $Interv$.*

We claim that a DTAPN N belonging to the class $\mathcal{N}_D(\mathbb{N}, Interv_3)$ can be reduced to a DTAPN N'_D in $\mathcal{N}_D(\mathbb{N}, Interv_1)$ preserving isomorphism. Since the only difference between the DTAPN model and the basic TAPN model is how time elapses, we can apply the same construction as in the proof of Theorem 3.1 to show that $\mathcal{N}_D(\mathbb{N}, Interv_3) \cong \mathcal{N}_D(\mathbb{N}, Interv_1)$. Thus, we can depict the hierarchy for classes of DTAPNs as shown in Figure 29 where the arrows lead from a more general class to a less general one.

The question of whether the hierarchy of the classes with continuous time is strict is still open.

6.1.2 Discrete Time

Finding a polynomial time reduction from DTAPNs with discrete time to TAPNs seems to be a difficult task. Since in DTAPNs we might have more than one clock, i.e. one for each equivalence class, where as for TAPNs we only have one global clock where all the tokens in the net grow older at once, the intuitive understanding of the problem is obvious.

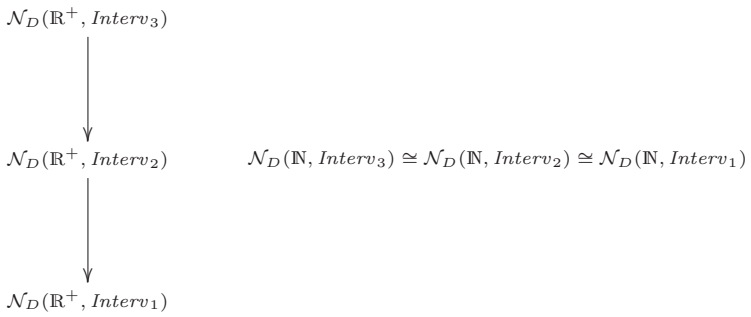


Figure 29: Class hierarchy for DTAPNs

We have tried to create a reduction in different ways, e.g. being inspired by [9], but always stumble upon this same problem. However, we will show that we can reduce discrete time DTAPNs to regular Petri nets (without time) preserving *timed-reachability*.

Definition 6.6.
Timed-reachability *is the problem, can we reach a marking M from an initial marking M_0 within at most v time units elapsed in every equivalence class? This is denoted by $M_0[\sigma\rangle_v M$ and formally we ask if $\exists\sigma$ s.t. $M_0[\sigma\rangle_v M$?*

We start by explaining the technique in [8] where TAPNs are polynomially reduced to PNs, following up with a section where we show how DTAPNs can be reduced to PNs applying the same technique, preserving timed-reachability.

Reducing TAPNs to PNs

In this section we implicitly assume the discrete time setting. We will explain the intuition behind the technique described in [8]. This reduction will help to answer the following problem:

$$\exists\sigma \text{ s.t. } M_0[\sigma\rangle_v M$$

This reduction preserves timed-reachability.

The first thing to notice is that in regular Petri nets no time related attributes are present, as in TAPNs, where time has a significant effect on the behavior of the net. It is therefore obvious that we need to find a way to represent time in ordinary PNs.

In [8] the authors suggest how to do this. By introducing special *clock* places and *tick* transitions to the Petri net, to play the role of a global clock, we can simulate the passage of time. Figure 30 shows an example of what we like to call a *time line*, where the clock places c_0, c_1 and c_2 are connected by the tick transitions $tick_1$ and $tick_2$.

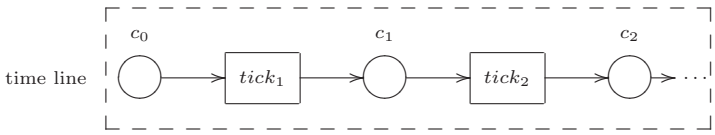


Figure 30: An example of a time line

At all times, we must have exactly one clock place marked in any given timeline. Initially we place a token in c_0, meaning 0 time has elapsed. The time line then indicates, in discrete steps, how long time has elapsed, by moving the token around. However, since time in TAPNs can elapse infinitely but PNs must have finitely many places and transitions, we can only reduce TAPNs to PNs up to a given instance in time v. So with this instance v we can reduce a given TAPN N to a PN N' up to it and the time line in N' will be finite.

Let us now look at how the time line helps us to represent time in an ordinary PN. As an example, we look at the simple TAPN N given in Figure 31. N has two places, p_1 and p_2 and one transition t_1 s.t. $In(p_1, t_1) = \{[1, 2]\}$ and $Out(t_1, p_2) = \{[0, 0]\}$.

We start the construction of a PN N' by creating the time line. In this example we set the time limit $v = 3$, so we create the clock places c_0, c_1, c_2 and c_3 and the tick transitions $tick_1$, $tick_2$ and $tick_3$.

Now we create, for each place p in N a set of places $\{p^0, p^1, \ldots, p^{v)}\}$ in N'. In our example, for p_1, we create the places p_1^0, p_1^1, p_1^2 and p_1^3 and for p_2 we create the places p_2^0, p_2^1, p_2^2 and p_2^3. See Figure 32.

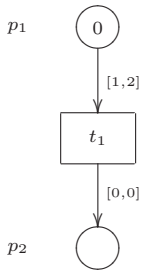


Figure 31: The TAPN N

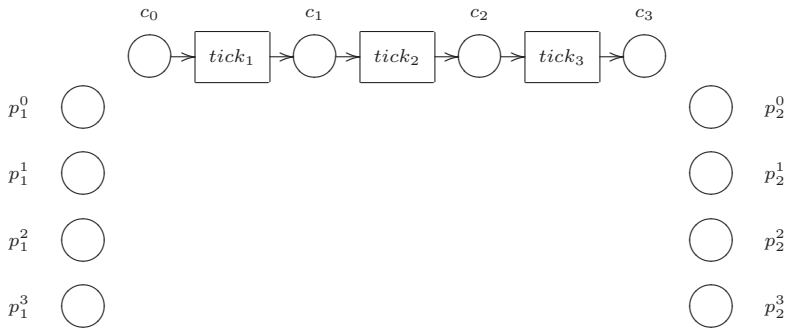
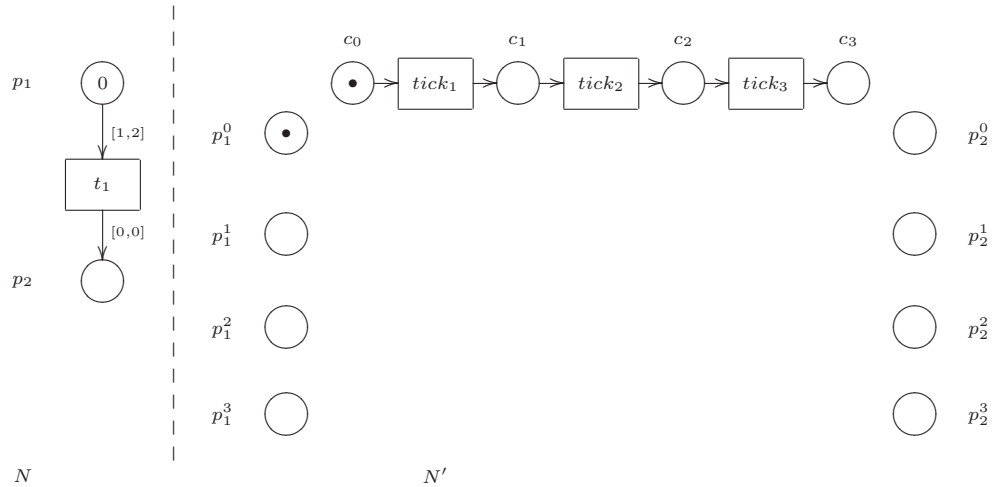
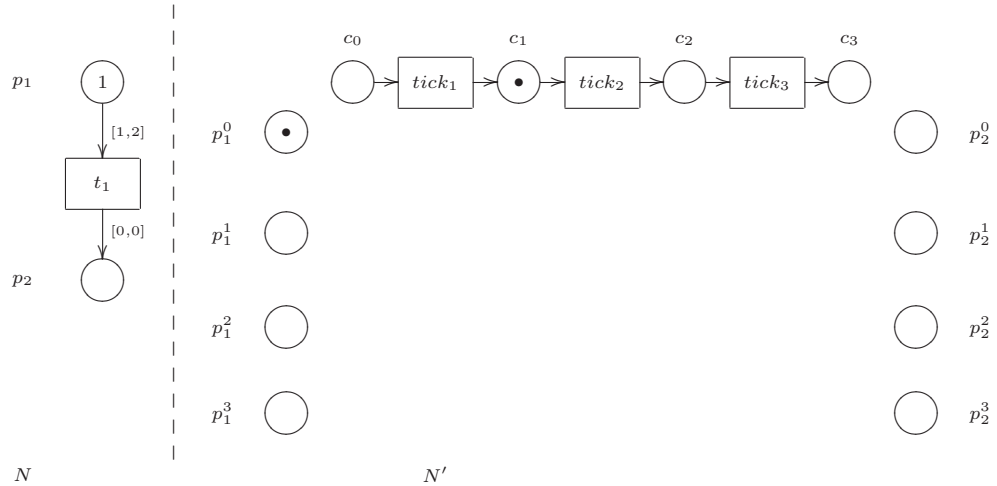


Figure 32: The net N'

The idea is that the ages of the tokens are controlled in a static way: tokens do not move when they become older, instead the clock places in the time line do the work. This way, we capture the age x of a token in place p in N by placing a token in one of the p^j places in N' s.t. for the marked clock place c_i, $i - j = x$. For example, look at how we represent the age of the token in p_1 in the TAPN N, by placing tokens in the PN N' in Figure 33.



(a)



(b)

Figure 33: Simulating time in a Petri net

In Figure 33 (a) the age of the token in p_1 in N, which is 0, is represented by placing tokens in p_1^0 and c_0 in N', i.e. marking places p^j and c_i represents that the age of the token in place p equals $i - j$, in this case p_1^0 and c_0 are marked in N', so they represent a token of age $0 - 0 = 0$ in place p_1 of N. In Figure 33 (b), one time unit has elapsed in N so that the age of the token in p_1 is 1. This is represented in N' by marking places p_1^0 and c_1(the purpose of the other p^j places will be explained later).

Now, let us move on to adding the transition t_1 to the picture. t_1 in N is enabled whenever we have a token in p_1 of age 1 or 2. The same has to be true for N', a transition representing t_1 in N' must be enabled when we a have a token in a place p_1^j and a clock place c_i s.t. $i - j = 1$ or $i - j = 2$. The *or* between these conditions implies that we need to have two transitions in N' that represent the one transition t_1 in N. In general, we create in N', for every transition t in N a set of transitions $\{t^n, \ldots, t^m\}$ where $\{n, \ldots, m\} = \{x \mid x \in In(p,t),\ x \in \mathbb{N},\ p \in {}^\bullet t\}$. In words, we create as many t^k transitions as there are values that fit the intervals from each input place of the transition t.

So for our transition t_1 in N, we create the transitions t_1^1 and t_1^2 in N', since $In(p_1, t_1) = \{[1, 2]\}$ (hence, $|\{x \mid x \in [1, 2]\}| = 2$).

As we said before, these transitions must be enabled when we a have a token in a place p_1^j and a clock place c_i s.t. $i - j = 1$ or $i - j = 2$ respectively. Thus, we put an arc from places p_1^0 and c_1 to t_1^1 in N', and from places p_1^0 and c_2 to t_1^2 in N'. We also have to add an arc from each of the transitions back to the time line, so we do not remove the token from the time line completely (these arcs were missing from the reduction given in [8]).

We are now well on our way in explaining the reduction from the TAPN N to the PN N'. We complete the picture by explaining the set of places $\{p_2^0, p_2^1, p_2^2, p_2^3\}$ in N' that represent the place p_2 in N (Figure 34).

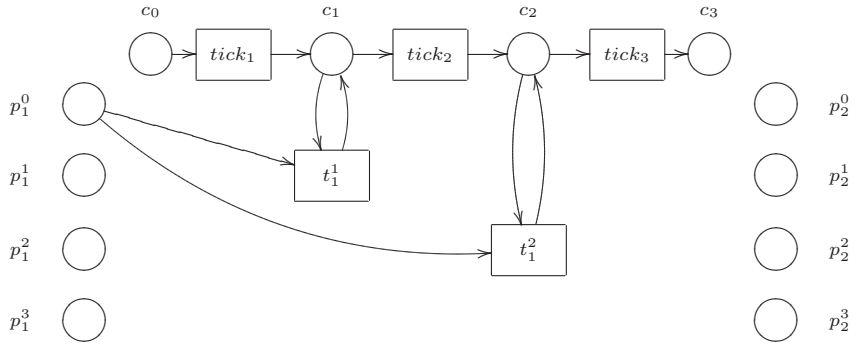


Figure 34: The net N'

Recall that the transition t_1 in N outputs a token to the place p_2 and $Out(t_1, p_2) = \{[0, 0]\}$, i.e. t_1 creates a token in p_2 of age 0. The same has to be true for the transitions that represent t_1 in N', namely t_1^1 and t_1^2. They

must both output a token to a place p_2^j, and its age should be represented as being 0. Lets start by looking at the t_1^1 transition. When it fires, it consumes a token from p_1^0 and c_1. This indicates, that since it should create a token of age 0 in one of the p_2^i places, that t_1^1 must output to the place p_2^1 (since the token in c_1 does not move as we fire t_1^1). However, the other transition t_1^2 consumes tokens from p_1^0 and c_2, so to create a token of age 0, t_1^2 must output to the place p_2^2. Figure 35 shows this.

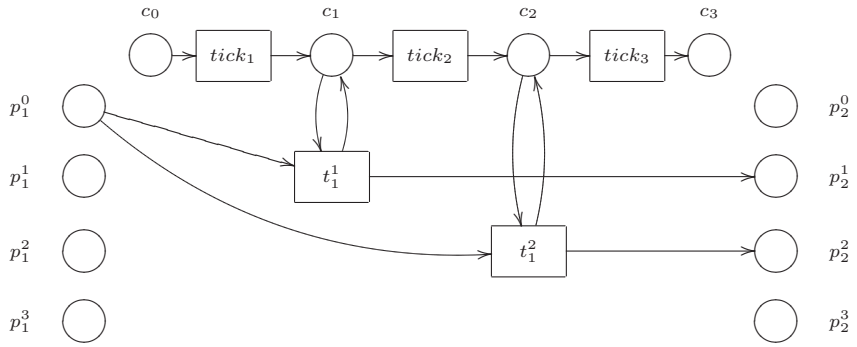


Figure 35: The net N'

In general, a transition t^k outputs to a place p^j in N' for all places p that are output places of t in N and $k - j \in Out(t, p) \cap \mathbb{N}$.

Now imagine that the TAPN N we have been working with is a part of a larger net (still with $v = 3$). We could easily be in the situation where time has elapsed for 1 time unit and we have just placed a token of age 0 in place p_1. Since time has elapsed for 1 time unit, the token in the time line is situated in place c_1, and to represent a token in p_1 of age 0, we have a token in p_1^1 (see Figure 36).

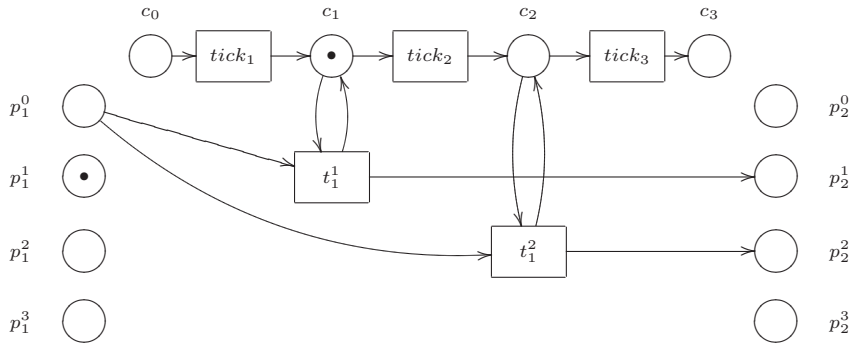


Figure 36: The net N'

From this situation, neither t_1^1 nor t_1^2 are enabled in N', and similarly t_1 is not enabled in N when a token of age 0 is in place p_1. However, we can let time elapse in N, making the token in p_1 grow to age 1, and then t_1 becomes enabled. The same must be true for N', if we let time elapse, i.e. fire the $tick_2$ transition, the token in p_1^1 represents a token of age 1, and then we should be able to fire a transition t_1^k.

Thus, we discover that only adding as many t^k transitions as there are values that fit the intervals from each place to the transition t, is not enough. We need more t_1^k transitions to represent t_1 in N'. Thus, we add a second index to the transitions in N' s.t. t_1^k becomes $t_1^{k,l}$ where $1 \leq l \leq v$.

Now we can add more transitions, $t_1^{k,l}$ to N', to represent t_1 consuming a token of age 1 from place p_1, after a total of 2 time units elapsing in the net (Figure 37). We add arcs to $t_1^{1,2}$ and $t_1^{2,2}$ from the corresponding input places and from $t_1^{1,2}$ and $t_1^{2,2}$ to the output places p_2^2 and p_2^3, to represent in N' the creation of a token of age 0 in place p_2, after t_1 has fired in N.

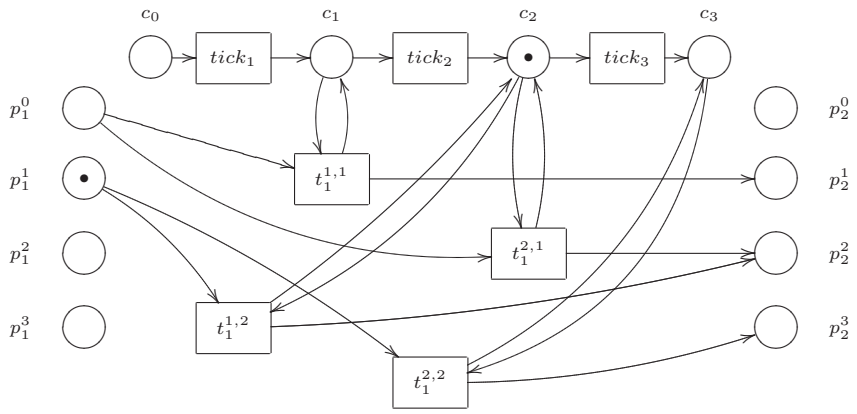


Figure 37: The net N'

Now we do the same for the places p_1^2 and p_1^3. We add the transitions $t_1^{1,3}$, $t_1^{2,3}$, $t_1^{1,4}$ and $t_1^{1,4}$ to complete the picture. We do not draw the transitions $t_1^{2,3}$, $t_1^{1,4}$ and $t_1^{1,4}$ since we are only looking at time elapsing up to v. The final picture of N' can be seen in Figure 38.

The formal definition and proof of this reduction can be found in [8].

Reducing DTAPNs to PNs

Recall that a TAPN is the same as a DTAPN, where all places belong to the same equivalence class. Therefore, we have now shown that we can reduce DTAPN where $\equiv = P \times P$ to PN. Let us look at the case where this does not hold i.e. we have more than one equivalence class. We are only handling the DTAPN model in discrete time setting.

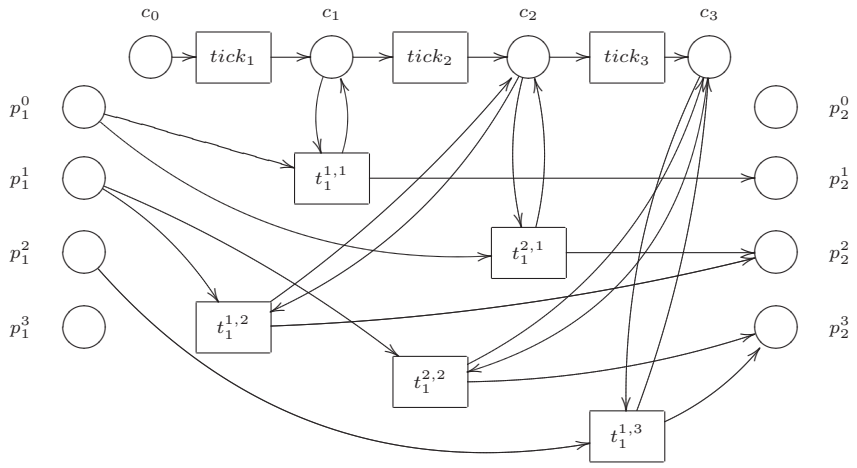


Figure 38: The net N'

We will show that we can polynomially reduce DTAPNs to PNs preserving timed-reachability.

As an example, we will use the DTAPN N in Figure 39, where place p_1 and p_2 are in separate equivalence classes, and the time constant $v = 3$.

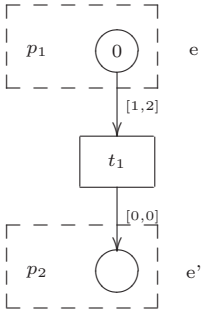


Figure 39: The DTAPN N

Now that we have two equivalence classes in N, we represent this in the PN N' by having two time lines, each representing how time elapses in its corresponding equivalence class. An element $e \in E$ is an equivalence class, s.t. $[p]_{\equiv} = [p']_{\equiv} \quad \forall p, p' \in e$. We create in N' two time lines, one for $e = \{p_1\}$ and one for $e' = \{p_2\}$, and places by the same rules as earlier, for each p in N we create a set of places $\{p^0, p^1, \ldots, p^k\}$ in N' (see Figure 40).

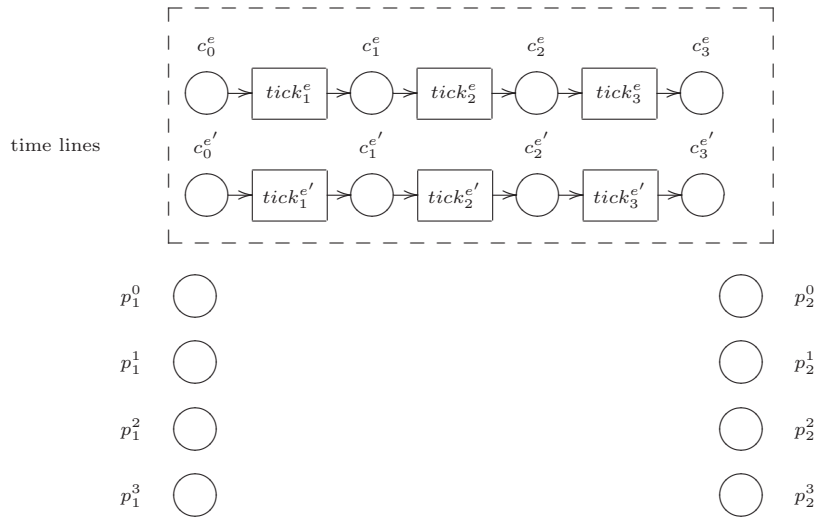


Figure 40: Creation of places in N'

Again, the idea is that the ages of the tokens in N' are controlled in a static way, the clock places in the time lines do the work. So to represent a token of age 0 in place $p \in e$ in N, we put a token in one of the p^j places in N' s.t. for the marked clock place c_i^e, $i - j = 0$. For example, look at how we represent the age of the token in p_1 in the DTAPN N, by placing tokens in the PN N' in Figure 41.

In Figure 41 (a) we represent two tokens of age 0, one in place p_1 and the other in p_2. We fire a time elapsing transition ϵ in N where $\epsilon(e) = 1$ and $\epsilon(e') = 0$, where the token in p_1 grows older by one time unit but the token in p_2 does not grow older. N' matches this transition firing by firing the $tick_1^e$ transition. Thus, in Figure 41 (b), N' represents the marking $\{(p_1, 1), (p_2, 0)\}$ of N.

We connect places to transitions in the same way as before, creating for each t in N, a set of transitions $\{t^{n,l}, \ldots, t^{m,l} | l \in \underline{v}^1\}$ where $\{n, \ldots, m\} = \{x \mid x \in In(p,t), x \in \mathbb{N}, p \in P\}$. In words, we create v times as many t^k transitions as there are values that fit the intervals from each place to the transition t.

So for our transition t_1 in N, we create $t_1^{1,1}$, $t_1^{2,1}$, $t_1^{1,2}$, $t_1^{2,2}$ and $t_1^{1,3}$ in N', since $In(p_1, t_1) = \{[1, 2]\}$ (hence, $|\{x \mid x \in [1, 2]\}| = 2$). This is shown in Figure 42.

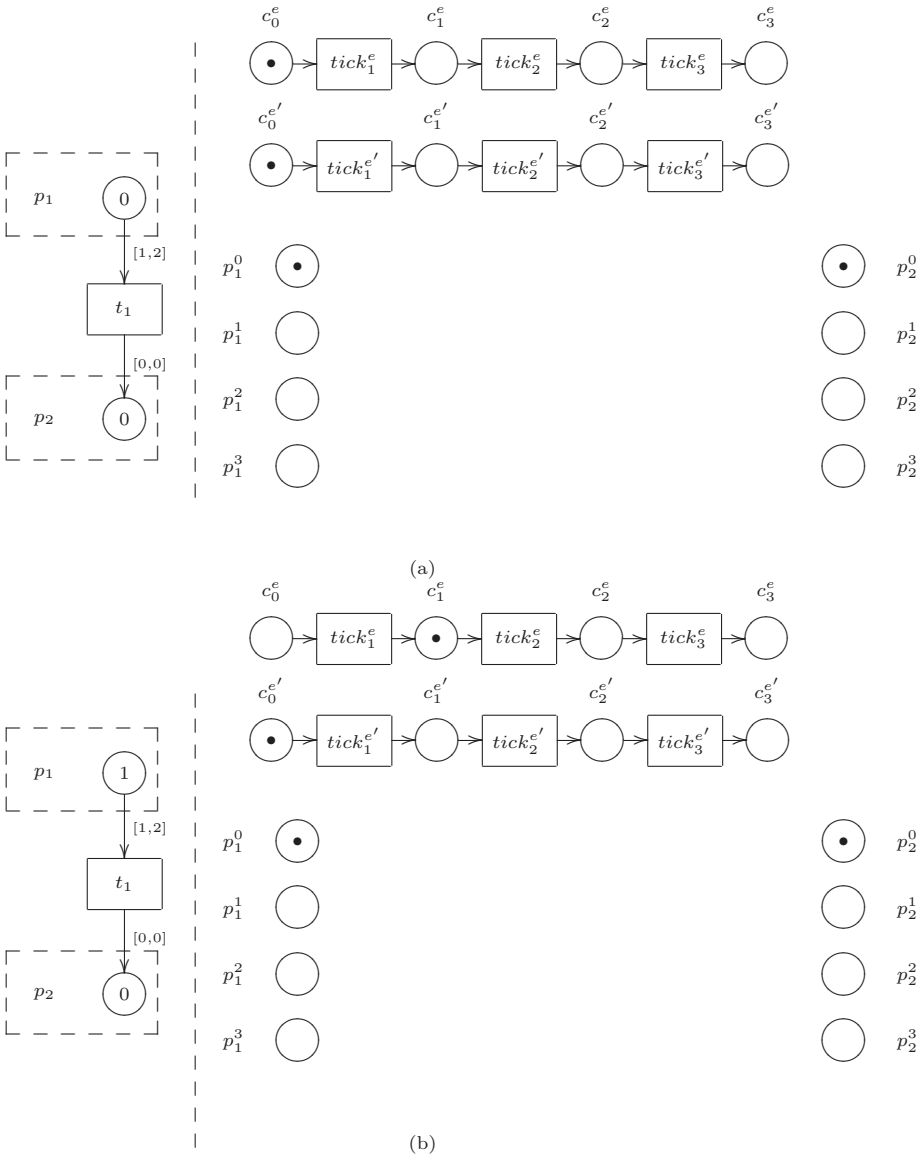


Figure 41: Simulation of time

Now, since t_1 outputs to p_2 in N, we need to add an arc from each of these $t_1^{k,l}$ transitions to one of the p_2^j places in N'. The token that is created must be of age 0, so we need to make sure that the token in place p_2^j is represented as

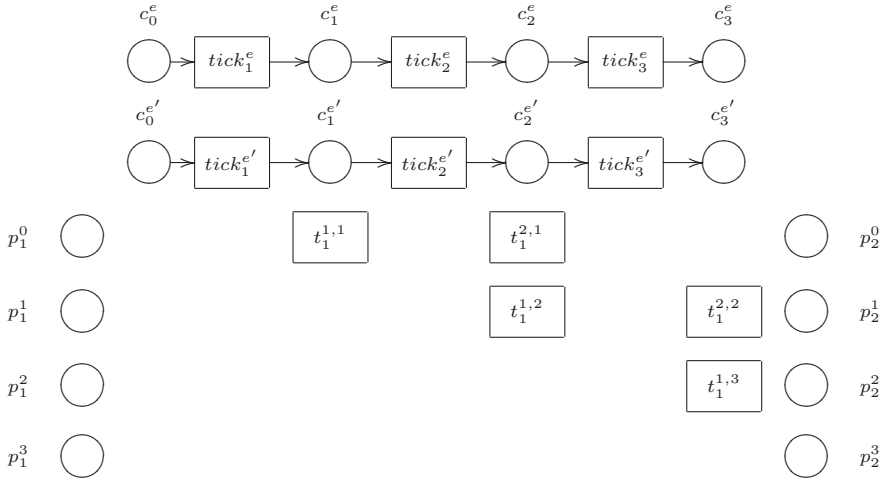


Figure 42: Creation of transitions in N'

being of age 0. Here this reduction gets even more complex, since the place p_2 is not in the same equivalence class as p_1, implying that the age of the tokens in the p_2^j places are represented by a different time line than the one that represents the ages of the tokens t_1 consumes.

Therefore, we need to duplicate the set of transitions in N' for every clock place in the time line of the equivalence class that the output place of t belongs to, and make sure that no matter where the token is in that time line, we always create a token of age 0 in p_2.

Let us look at our example again. In N, t_1 outputs to p_2, which does not belong to the same equivalence class as p_1, the input place of t_1. Therefore, we need to add more transitions and an arc from each transition to an output place, where each transitions checks where the token is in the time line for the equivalence class of p_2. So, for each $t_i^{k,l}$ transition we create transitions $\{t^{k,l,1}, \ldots, t^{k,l,|E|}\}$. In Figure 43 it is shown how the duplicates of transitions $t_1^{1,1}$ will look like and where they will output.

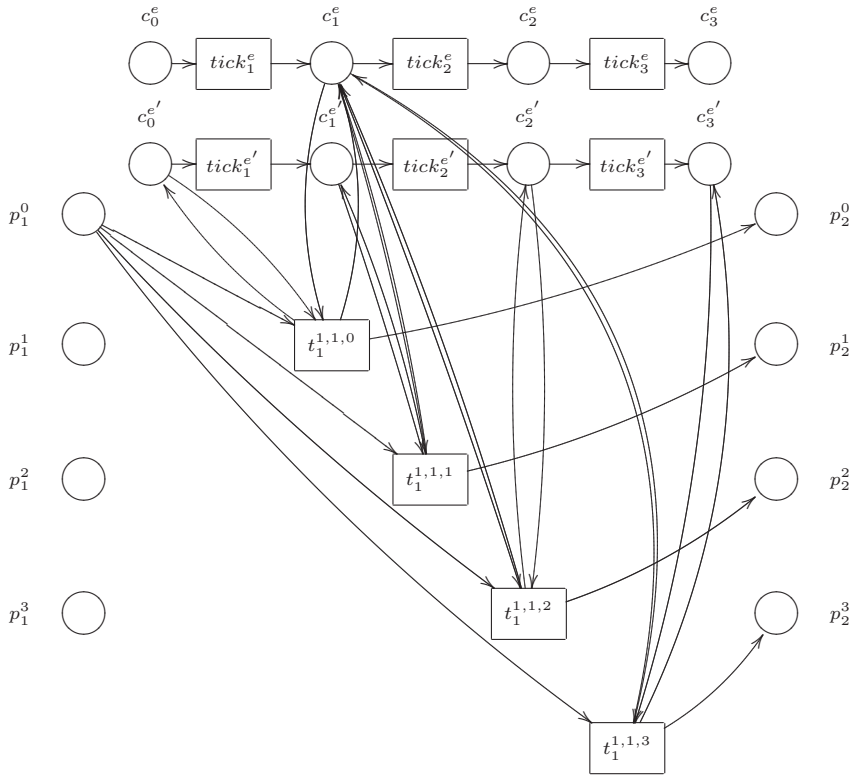


Figure 43: Duplicate transitions in N'

Formally, we define this reduction as follows:

Definition 6.7.
Given a DTAPN $N = (P, T, In, Out, \equiv)$, in the class $\mathcal{N}_D(\mathbb{N}, Interv_1)$, with initial marking M_0, we define the associated Petri net $N' = (P', T', In', Out')$ as follows.

Let $F = \{(p, ppos, cpos) | p \in P, ppos \in \underline{v}^0, cpos \in \underline{v}^0\}$ be a multiset.

- We say that a multiset $F = \{(p_1, ppos_1, cpos_1), \ldots, (p_n, ppos_n, cpos_n)\}$ instantiates a multiset $\{(p_1, \mathcal{I}_1), \ldots, (p_n, \mathcal{I}_n)\}$ where $\mathcal{I}_i \in Interv_1$ if and only if $(cpos_i - ppos_i) \in \mathcal{I}_i$ for all $i \in \underline{n}^1$.

- $P' = \{p^j | p \in P, j \in \underline{v}^0\} \cup \{c_i^e | i \in \underline{v}^0, e \in E(\equiv)\}$
- $T' = \{t(F, F') | t \in T,$ *F instantiates $In(t)$ and F' instantiates $Out(t)$*$\} \cup \{tick_i^e | i \in \underline{v}^1, e \in E(\equiv)\}$.

- $In'(c^e_{i-1}, tick^e_i) = 1 \ \forall i \in \underline{v}^1 \ \forall e \in E(\equiv)$

  $In'(c^e_i, t(F,F')) = \begin{cases} 1 & \text{if } \exists (p, ppos, i) \in F \text{ s.t. } p \in e \\ 0 & \text{otherwise} \end{cases}$

  $In'(p^j, t(F,F')) = \sum_{cpos=0}^{v} F(p, j, cpos)$

- $Out'(tick^e_i, c^e_i) = 1 \ \forall i \in \underline{v}^1 \ \forall e \in E(\equiv)$

  $Out'(t(F,F'), c^e_i) = \begin{cases} 1 & \text{if } \exists (p, ppos, i) \in F' \text{ s.t. } p \in e \\ 0 & \text{otherwise} \end{cases}$

  $Out'(t(F,F'), p^j) = \sum_{cpos=0}^{v} F'(p, j, cpos)$

Finally, a marking M on N can be translated to a set of markings $f(M)$ on N' as follows:

We define the inverse function $f^{-1}(M') = M$. Let $clock_p = i$ where $p \in e$ and $M'(c^e_i) = 1$. Then:

$$M(p) = \bigcup_{j=0}^{v} \{u_1, \ldots, u_n | n = M'(p^j), u_i = clock_p - j \ \text{ for } 1 \leq i \leq M'(p^j)\}$$

Thus, $f(M) = \{M' | f^{-1}(M') = M\}$.

A general picture of the reduction is presented in Figure 44 where ${}^\bullet t = \{p_1, \ldots, p_n\}$ and $t^\bullet = \{q_1, \ldots, q_m\}$.

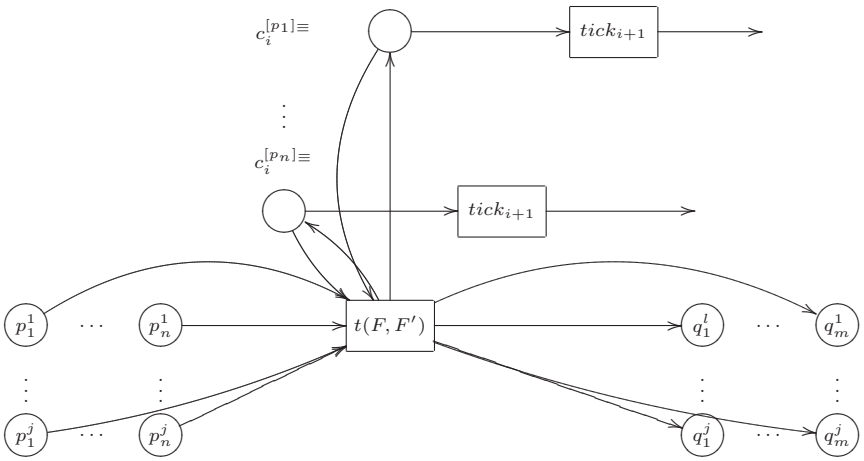


Figure 44: Reducing DTAPN to PN

Theorem 6.1.
Given a DTAPN $N \in (\mathbb{N}, Interv_1)$ with an initial marking M_0, a marking M

on it and a constant v, we can reduce it, in polynomial time, to a corresponding PN N' preserving timed reachability,i.e.

$$M_0[\sigma\rangle_v M \text{ if and only if } \exists M_0' \in f(M_0).\ \exists M' \in f(M).\ \exists \sigma' \text{ s.t. } M_0'[\sigma'\rangle M'$$

Proof.
Assume that we can reach M from M_0 within v time units, i.e. there exists a $\sigma = t_1, ..., t_n$ s.t. $M_0[\sigma\rangle_v M$. We show how firing of a transition t from σ can be simulated in N':

- $t \in T$:
  When t fires in N we consume a set of tokens $\{x_1, \ldots, x_f\} \subseteq M(p)$ from each place $p \in {}^\bullet t$ and create a set of tokens $\{y_1, \ldots, y_g\}$ in each place $p \in t^\bullet$. To match this transition firing, we fire a transition $t(F, F')$ in N' where F corresponds to the set $In(t)$ and F' corresponds to the set $Out(t)$, and $t(F, F')$ simulates the firing of $t(F, F')$ in the PN N'.

- $\epsilon \in [E(\equiv) \to \mathbb{N}]$:
  When a time elapsing transition ϵ fires in N, we match it by firing a sequence of *tick* transitions in N' for each $e \in E(\equiv)$, i.e. we fire $tick^e_{r+1}$, $tick^e_{r+2}$, $\ldots$, $tick^e_s$ where the clock place c^e_r is marked in N' and $\epsilon(e) = s$.

Assume that M cannot be reached from M_0 within v time units, i.e. there does not exists a $\sigma = t_1, ..., t_n$ s.t. $M_0[\sigma\rangle_v M$. Then, for any $M_0' \in f(M_0)$ and $M' \in f(M)$, there does not exists $\sigma' = t_1, \ldots, t_m$ for N' s.t. $M_0'[\sigma'\rangle M'$. □

Corollary 6.1 follows since $\mathcal{N}_D(\mathbb{N}, Interv_3) \cong \mathcal{N}_D(\mathbb{N}, Interv_1)$.

Corollary 6.1.
Given a DTAPN N in $\mathcal{N}_D(\mathbb{N}, Interv_3)$ with an initial marking M_0, a marking M on it and a constant v, we can reduce it to a corresponding PN N' preserving timed reachability,i.e.

$$M_0[\sigma\rangle_v M \text{ if and only if } \exists M_0' \in f(M_0).\ \exists M' \in f(M).\ \exists \sigma' \text{ s.t. and } M_0'[\sigma'\rangle M'$$

Showing a stronger equivalence relation is still an open problem. We believe that DTAPNs in $\mathcal{N}_D(\mathbb{N}, Interv_3)$ are weakly bisimilar to PNs up to a certain "depth", i.e. we can reduce DTAPNs to PNs up to a given time constant v.

6.1.3 Continuous Time

We now consider the DTAPN model in continuous time. We apply the same technique as we did for the TAPN model, backwards reachability analysis using existential zones.

We want to determine, given a DTAPN N in $\mathcal{N}_D(\mathbb{R}^+, Intervv_3)$ with an initial marking M_0 and a marking M, if we can reach a marking M' from M_0 s.t. $M \subseteq M'$. As before, we start by defining a zone Z that characterizes the marking M and compute the transitive closure $pre^*(Z)$.

To calculate the set $pre(Z)$ we need to compute two things, $pre_t(Z)$ and $pre_\epsilon(Z)$. We calculate $pre_t(Z)$ using the same formula as for regular TAPNs from Section 5.3. We need to redefine $pre_\epsilon(Z)$ for the DTAPN model, where we, in addition to lowering the lower bounds of each token to 0, relax the time constraints between tokens that are in places that belong to different equivalence classes. The set $pre_\epsilon(Z)$ is the largest zone that satisfies $\forall M \in [\![Z]\!]$. $\exists M' \in [\![Z']\!]$. $\exists \epsilon$ *s.t.* $M[\epsilon\rangle M'$. (Figure 45).

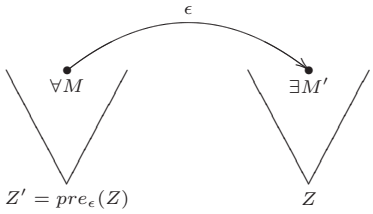


Figure 45: Existential zones

For a given zone Z we calculate $pre_\epsilon(Z)$ as shown in Definition 6.8.

Definition 6.8.
For a normalized existential zone $Z = (m, \overline{P}, D)$, $pre_\epsilon(Z)$ is the existential zone $Z' = (m, \overline{P}, D')$ where:

- $D'(0, j) = 0$ *for* $j \in \underline{m}^1$
- $D'(i, 0) = D(i, 0)$ *for* $i \in \underline{m}^1$
- $D'(i, j) = \begin{cases} D(i,j) & \text{if } \overline{P}(x_i) \equiv \overline{P}(x_j) \\ \infty & \text{if } \overline{P}(x_i) \not\equiv \overline{P}(x_j) \end{cases}$ *for* $i \in \underline{m}^1$, $j \in \underline{m}^1$

Theorem 6.2.
For a normalized existential zone Z, $pre_\epsilon(Z) = Z'$ is the largest zone that satisfies: $\forall M \in [\![Z']\!]$. $\exists M' \in [\![Z]\!]$. $\exists \epsilon$ s.t. $M[\epsilon\rangle M'$ for some ϵ.

The following proof is a slight modification of [2]:

Proof.
All zones are assumed to be normalized.

Assume that a marking $M' \models Z$ and $\exists \epsilon$ s.t. $M[\epsilon\rangle M'$. We show that $M \models Z' = pre_\epsilon(Z)$:

- M satisfies the same constraints on clock differences for places belonging to the same equivalence class as M' since all the clocks are increased by the same value. Other difference constraints are released and trivially satisfied.
- M' satisfies the upper bounds of single clock values in Z, and therefore M will satisfy the (same) upper bounds in Z'.
- Z' has no lower bounds on single clock values, so they are trivially satisfied by M since time elapses backwards.

Assume that a marking $M \models Z' = pre_\epsilon(Z)$. We show that $M[\epsilon\rangle M'$ for some ϵ and $M' \models Z$.

We define ϵ to be the minimum time delay required so that each token in M' will satisfy the lower bound required by Z. The normality of Z and the minimality of ϵ ensure that the delay will not make any token larger than the corresponding upper bound in Z. □

Corollary 6.2 follows:

Corollary 6.2.
Coverability is decidable for DTAPNs in the class $\mathcal{N}_D(\mathbb{R}^+, Interv_3)$ using backwards reachability analysis and existential zones.

6.1.4 Summary

We have examined the decidability of properties for DTAPNs. In [15] the authors claim that reachability is undecidable for all classes of the DTAPN model. They also claim that coverability is decidable for DTAPNs in $\mathcal{N}_D(\mathbb{N}, Interv_1)$ and $\mathcal{N}_D(\mathbb{R}^+, Interv_1)$ and we have proofed this to be true.

We have further examined the decidability of coverability and shown that coverability is decidable for all classes of DTAPNs. We also proved that timed-reachability is decidable for all classes of DTAPNs with discrete time, whereas this is still an open problem for the classes of DTAPNs with continuous time.

It is also worth mentioning that since we have shown that both the reachability problem and the deadlock problem are undecidable for TAPNs, it follows that they are also undecidable for this more general model.

6.2 Age-Preserving TAPN

By definition, transitions in timed-arc Petri nets consume tokens whose ages lie within some interval, and create new tokens whose ages nondeterministically lie within another (possibly identical) interval.

We define a new model based on the TAPN model, where a new kind of transitions, called *age-preserving transitions* are allowed. When an age-preserving transition is fired, the tokens created in the output places are of the same age as the token that the transition consumed from its input place. Graphically, age-preserving transitions are represented by a box with a double border.

6.2.1 The Age-Preserving Model

We define the TAPN model with age-preserving transitions. Let us fix the interval set $Interv \in \{Interv_1, Interv_2, Interv_3\}$ and the time domain $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$.

Definition 6.9.
*A **timed-arc Petri net (TAPN) with age-preserving transitions** is a 4-tuple $N = (P, \mathbb{T}, In, Out)$ where*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places,*
- $\mathbb{T} = T \cup \hat{T}$ *where* $T = \{t_1, t_2, ..., t_m\}$ *and* $\hat{T} = \{\hat{t_1}, \hat{t_2}, ...\hat{t_k}, \}$ *are finite sets of regular and age-preserving transitions respectively s.t.* $\hat{T} \cap T = \emptyset$ *and* $P \cap \mathbb{T} = \emptyset$*,*
- $In : P \times \mathbb{T} \to Interv^{\oplus}$*, is a function that associates intervals to arcs* (p, t) *and* $(p, \hat{t})$*. However, we restrict the number of input places to transitions* $\hat{t} \in \hat{T}$ *to be equal to one, s.t.* $\sum_{p \in P} |In(p, \hat{t})| = 1$.
- $Out : T \times P \to Interv^{\oplus}$*, is a function that associates intervals to arcs of the form* (t, p)*.*
  $Out : \hat{T} \times P \to \mathbb{N}$ *associates a natural number to* $(\hat{t}, p)$*, the number of arcs between* $\hat{t}$ *and* p*. The time intervals are irrelevant, since output arcs from an age-preserving transition do not use them.*

We limit the number of input places that an age-preserving transition can have, an age-preserving transition has to have exactly one input place. Should we allow less than one input place, we would not be able to tell the ages of the tokens created in the transitions output places. Should we allow more than one input place to age-preserving transitions, the model would be somewhat more complicated. We would have to follow some method of determining the ages of the tokens that are to be created in the output places, and we would not always be able to make the transition preserve the age of each consumed token (since we are not guaranteed that the number of output places are the same as the number of input places) and thus, this kind of transition would belong to a different model.

We use the notation ${}^\bullet\hat{t} = \{p \mid In(p,\hat{t}) \neq \emptyset\}$ to denote the multiset of all input places of transition t and similarly we use $\hat{t}^\bullet = \{p \mid Out(\hat{t},p) \neq 0\}$ to denote the multiset of all output places of t.

The firing rule for TAPN model with age-preserving transitions is identical to the firing rule in Definition 6.20 for $t \in T$. For $\hat{t} \in \hat{T}$ we have the following definition.

Definition 6.10.
Let $N = (P, \mathbb{T}, In, Out)$ be a TAPN with age-preserving transitions, M a marking on it and $\hat{t} \in \hat{T}$, where $In(\hat{t}) = \{(p, \mathcal{I})\}$.
*We say that $\hat{t}$ is **enabled** at marking M if and only if:*

$$\exists x \in \mathcal{I} \cap M(p)$$

i.e. in the input place, there exists at least one token whose age fits the interval of the input arc leading from p to $\hat{t}$.
Let us fix x s.t. $x \in \mathcal{I} \cap M(p)$. Then we can define:

- $X_p = \{x\}$
- $X_{p'} = \emptyset \ \forall p' \neq p$
- $Y_{p'} = \begin{cases} \{x, x, \ldots, x\} \text{ where } x \text{ occurs } Out(\hat{t},p') \text{ times} & \text{if } p' \in \hat{t}^\bullet \\ \emptyset & \text{otherwise} \end{cases}$

*Now we can define the **firing rule** for $\hat{t} \in \hat{T}$:*

$$M'(p) = (M(p) \setminus X_p) \cup Y_p \quad \forall p \in P$$

Notice that as the regular t transitions, an enabled age-preserving transition $\hat{t}$ can fire, but it does not have to. We model the passage of time for this model the same way we did for the basic TAPN model, Definition 3.22.

Classes of Age-Preserving TAPNs
The definitions above provide a general way of defining the time attributes of the age-preserving TAPN model, both the types of intervals and the time domain. We define classes of age-preserving TAPNs as follows.

Definition 6.11.
*Let $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ be a time-domain and $Interv \in \{Interv_1, Interv_2, Interv_3\}$ be the set of allowed intervals. Then we define $\mathcal{N}_A(\mathcal{D}, Interv)$ as the **class of all age-preserving TAPNs** with time domain $\mathcal{D}$ and intervals set $Interv$.*

We claim that an age-preserving TAPN N belonging to the class $\mathcal{N}_A(\mathbb{N}, Interv_3)$ can be reduced to an age-preserving TAPN N' in $\mathcal{N}_A(\mathbb{N}, Interv_1)$ preserving isomorphism. This can be shown by applying the same technique as in Section

3.2.1 when reducing TAPNs in $\mathcal{N}(\mathbb{N}, Interv_3)$ to TAPNs in $\mathcal{N}(\mathbb{N}, Interv_1)$. Thus, we can depict the hierarchy for classes of age-preserving TAPNs as shown in Figure 46 where the arrows lead from a more general class to a less general one. The strictness of the hierarchy remains an open question.

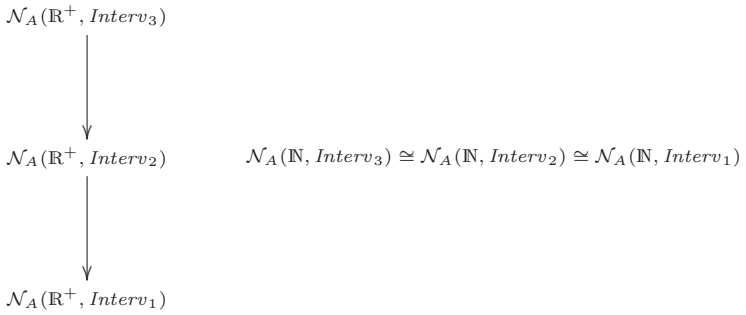


Figure 46: Class hierarchy for age-preserving TAPNs

6.2.2 Discrete Time

We show how we can build an ordinary timed-arc Petri net in the class $\mathcal{N}(\mathbb{N}, Interv_1)$, that simulates an age-preserving TAPN in $\mathcal{N}_A(\mathbb{N}, Interv_1)$, preserving strong bisimilarity.
Remark:
Since we have shown that $\mathcal{N}_A(\mathbb{N}, Interv_3) \cong \mathcal{N}_A(\mathbb{N}, Interv_1)$ and $\mathcal{N}(\mathbb{N}, Interv_3) \cong \mathcal{N}(\mathbb{N}, Interv_1)$, this reduction applies to all classes of age-preserving TAPNs and TAPNs with discrete time domain.

Inspired by the technique described in [8] we define a *maximal guard* of an age-preserving TAPN with age-preserving transitions:

Definition 6.12.
Let $N = (P, \mathbb{T}, In, Out)$ in $\mathcal{N}_A(\mathbb{N}, Interv_3)$ be an age-preserving TAPN, and $\mathcal{X} = [\![a, b]\!]$ for some $a \in \mathbb{N}$ $b \in \mathbb{N} \cup \{\infty\}$. We define limit : $\mathcal{I} \to \mathbb{N}$ as a function:

$$limit(\mathcal{X}) = \begin{cases} a & \text{if } b = \infty \\ b & \text{if } b \neq \infty \end{cases}$$

We define the set of all intervals in an age-preserving TAPN N,

$$all_N = \bigcup_{p \in P, t \in \mathbb{T}} In(p,t) \bigcup_{p \in P, t \in T} Out(t,p).$$

*The **maximal guard** of an age-preserving TAPN N can now be defined as follows:*

$$maxGuard(N) = max\{limit(\mathcal{I}) \mid \mathcal{I} \in all_N\}.$$

Let $G(N) = maxGuard(N) + 1$ in the following definitions.

The intuitive idea behind this simulation, is to capture the behavior of an age-preserving transition $\hat{t}$ by a set of regular transitions.

Figure 47 shows an example where the age-preserving transition $\hat{t}_1$ in N, where $G(N) = 3$, is simulated by the set of regular transitions, $\{t_1^1, t_1^2, t_1^3\}$ in N':

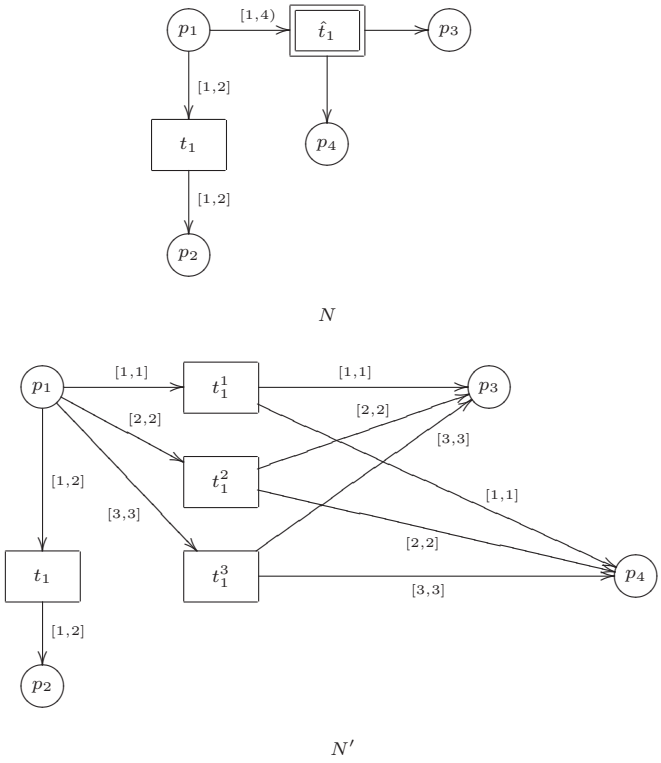
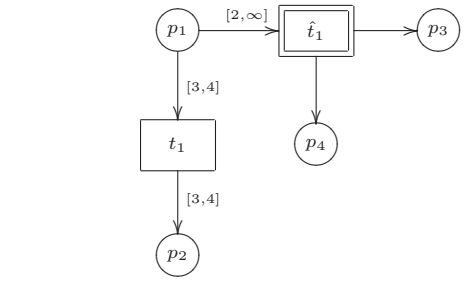


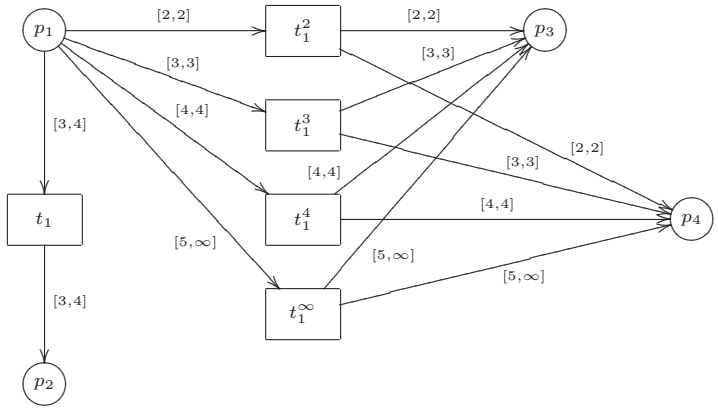
Figure 47: Example of a simulation

This simulation is straight forward when the ∞ sign is not present in the net. Otherwise we add a special transition to the set, i.e. to simulate an age preserving transition $\hat{t}$ in a net where the ∞ sign is present, we capture its behavior by the set of transitions $\{t^a, \ldots, t^b, t^\infty\}$, where $b = G(N) - 1$ and $In(p, \hat{t}) = \{[a, b]\}$ for $p \in {}^\bullet\hat{t}$. For example in Figure 48, the age-preserving transition $\hat{t}_1$ in N where $G(N) = 5$, is simulated by the set of transitions $\{t_1^2, t_1^3, t_1^4, t_1^\infty\}$ in N'.

Definition 6.13.
*A **labeled TAPN with age-preserving transitions**, in the class $\mathcal{N}_A(Interv, \mathcal{D})$, is a 6-tuple $\mathcal{L}(N) = (P, \mathbb{T}, In, Out, \Lambda, \lambda)$ where $(P, \mathbb{T}, In, Out)$ is a age-preserving TAPN and*



N



N'

Figure 48: Example of a simulation, $G(N) = 5$

- Λ *is finite set of labels*
- $\lambda : \mathbb{T} \to \Lambda$ *is a labeling function that gives each transition in the age-preserving TAPN a label*

Theorem 6.3.
Given an age-preserving TAPN N in the class $\mathcal{N}_A(\mathbb{N}, Interv_1)$ we can construct a regular TAPN N' in the class $\mathcal{N}(\mathbb{N}, Interv_1)$ that simulates the behavior of N s.t. $T(N') \sim T(N)$.

Proof.
Given an age-preserving TAPN $N = (P, \mathbb{T}, In, Out, \Lambda, \lambda)$ where $\mathbb{T} = \hat{T} \cup T$, $\hat{T} \cap T = \emptyset$ and the initial marking is M_0, we construct a TAPN $N' = (P', T', In', Out', \Lambda', \lambda')$ with initial marking $M_0' = M_0$ s.t.

- $P' = P$
- $T' = \{t \mid t \in T\} \cup$
  $\{t^i \mid \hat{t} \in \hat{T},\ a \le i \le b \text{ where } In(p, \hat{t}) = \{\!\{a, b\}\!\},\ a, b \in \mathbb{N},\ p \in P\} \cup$

$\{t^i \mid \hat{t} \in \hat{T},\ a \le i \le G(N)$ *where* $In(p,\hat{t}) = \{\!\![a,\infty]\!\!]\},\ a \in \mathbb{N},\ p \in P\} \cup$
$\{t^\infty \mid \hat{t} \in \hat{T},\ In(p,\hat{t}) = \{\!\![a,\infty]\!\!]\},\ a \in \mathbb{N},\ p \in P\}$

- $In'(p,t) = \begin{cases} In(p,t) & \text{if } t \in T,\ \forall p \in P \\ \{[i,i]\} & \text{if } t = t'^i \in T' \text{ and } i < \infty,\ \forall p \in {}^\bullet t'^i \\ \{[G(N),\infty]\} & \text{if } t = t'^\infty \in T',\ \forall p \in {}^\bullet t'^i \\ \emptyset & \text{otherwise} \end{cases} \quad \forall t \in T'$

- $Out'(t,p) = \begin{cases} Out(p,t) & \text{if } t \in T,\ \forall p \in P \\ \{[i,i]\} & \text{if } t = t'^i \in T' \text{ and } i < \infty,\ \forall p \in {}^\bullet t'^i \\ \{[G(N),\infty]\} & \text{if } t = t'^\infty \in T',\ \forall p \in {}^\bullet t'^i \\ \emptyset & \text{otherwise} \end{cases} \quad \forall t \in T'$

- $\Lambda' = \Lambda$

- $\lambda'(t) = \begin{cases} \lambda(t) & \text{if } t \in T \\ \lambda(\hat{t}') & \text{if } t = t'^i \in T' \end{cases}$

We show that $T(N') \sim T(N)$:

We define a strong bisimulation $\mathcal{R}$. For any markings M_1' and M_1 on $T(N')$ and $T(N)$ respectively, $(M_1', M_1) \in \mathcal{R}$ if and only if:

- $\forall p \in P'.\ |M_1'(p)| = |M_1(p)|$
- $\forall p \in P'.\ \{x \mid x \in M_1'(p),\ x < G(N)\} = \{x \mid x \in M_1(p),\ x < G(N)\}$

Should an age-preserving transition $\hat{t} \in \hat{T}$ fire in N, we consume a token of a given age x from the input place of $\hat{t}$ and create tokens of the same age in its output places. The ordinary TAPN N' can behave in the same way, where we fire a corresponding transition $t^i \in T'$ if the age of the token is less than $G(N)$, i.e. $x < G(N)$.

However, if the age of the token is greater than or equal to $G(N)$, i.e. $x \ge G(N)$, we fire a transition $t^\infty \in T'$, that consumes a token of any age greater than $G(N)$ and creates tokens of any age greater than $G(N)$.

Creation of tokens that are not necessarily of the same age as the token consumed in this case does not matter. The behavior of N' is still the same as the behavior of N, no matter how old the tokens are, as long as they are older than the maximal guard (tokens older than the maximal guard can only be used to fire transitions where the input intervals are of the form $[\![a,\infty]\!]$).

Thus, it is obvious that whenever $(M_1', M_1) \in \mathcal{R}$ then $\forall \alpha \in \Lambda \cup \{\epsilon(n) | n \in \mathbb{N}\}$,

- if $M_1' \xrightarrow{\alpha} M_2'$ in N' then $M_1 \xrightarrow{\alpha} M_2$ in N and $(M_2', M_2) \in \mathcal{R}$
- if $M_1 \xrightarrow{\alpha} M_2$ in N then $M_1' \xrightarrow{\alpha} M_2'$ in N' and $(M_2', M_2) \in \mathcal{R}$

and $T(N') \sim T(N)$, because the initial markings belong to $\mathcal{R}$. $\square$

We have shown that the age-preserving model in discrete time is equivalent to the basic TAPN model, and thus, we can translate the decidability results from the basic TAPN model to the age-preserving model in discrete time. Next section examines the age-preserving TAPN model in continuous time.

6.2.3 Continuous Time

Age-preserving TAPNs in the class $\mathcal{N}_A(\mathbb{R}^+, Interv_3)$ do not appear to be easily reduced to the basic TAPN model. The reduction given for age-preserving TAPNs with discrete time domain does not apply, since we would have to create infinitely many transitions in the TAPN. However, we show that we can apply backwards reachability and existential zones to decide coverability for the age-preserving TAPN model.

We start by giving a definition of how we apply the $\Cap$ function to a set of intervals:

Definition 6.14.
$\Cap$ is a function where, for some set of intervals $\mathbb{K} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\} \subseteq Interv_3$, $\Cap\mathbb{K} \in Interv_3$ and $\Cap\mathbb{K}$ is the largest possible interval s.t. $\Cap\mathbb{K} \subseteq \mathcal{K}$ for all $\mathcal{K} \in \mathbb{K}$

So for example, given a set of intervals $K = \{[1,6],[5,7],(5,9]\}$, $\Cap K = (5,6]$, the largest possible interval from K.

We can apply backwards reachability and existential zones to the age-preserving model by adding a small part to the definitions given in Section 5. But first, let us define the function *range*:

Definition 6.15.
*For a given difference bound matrix D and an index i, **range** is a function that returns the upper and lower constraints stored in D, for the artificial token represented in D with index i, i.e.*

$$range(i) = \begin{cases} [-D(0,i), D(i,0)] & \text{if } D(0,i) \in \mathbb{Z} \text{ and } D(i,0) \in \mathbb{Z} \\ (-D(0,i), D(i,0)] & \text{if } D(0,i) \in \overline{\mathbb{Z}} \text{ and } D(i,0) \in \mathbb{Z} \\ [-D(0,i), D(i,0)) & \text{if } D(0,i) \in \mathbb{Z} \text{ and } D(i,0) \in \overline{\mathbb{Z}} \\ (-D(0,i), D(i,0)) & \text{if } D(0,i) \in \overline{\mathbb{Z}} \text{ and } D(i,0) \in \overline{\mathbb{Z}} \end{cases}$$

In Section 5 we calculate $pre(Z)$ of a given normalized zone Z by tracing back all possible transitions in a given TAPN. For the age-preserving model we do the same, and now we also need to consider the age-preserving transitions, $\hat{t} \in \hat{T}$. Definition 6.16 shows how we calculate $pre_{\hat{t}}(Z)$ for a given zone Z.

Definition 6.16.
Consider an age-preserving TAPN $N = (P, \mathbb{T}, In, Out)$, a transition $\hat{t} \in \mathbb{T}$, and an existential zone $Z = (m, \overline{P}, D)$. Let $In(\hat{t}) = \{(p, \mathcal{I})\}$ and $\hat{t}^\bullet = \{q_1, \ldots, q_\ell\}$.

Then $pre_{\hat{t}}(Z)$ is the smallest set containing each existential zone Z' such that there is a partial injection $h : \underline{m}^1 \to \underline{\ell}^1$ with domain $\{i_1, ..., i_n\}$ satisfying the following conditions:

(1) $\overline{P}(i) = q_{h(i)}, \forall i \in \{i_1, \ldots, i_n\}$
(2) $Z \otimes (\mathcal{I}, i_1) \otimes ... \otimes (\mathcal{I}, i_n)$ *is consistent*
(3) $Z' = Z/i_1/.../i_n \oplus (p, \uplus(\{range(i)|i \in \{i_1, \ldots i_n\}\} \cup \mathcal{I}))$

So, to calculate a zone $Z' \in pre_{\hat{t}}(Z)$ we start by mapping the artificial tokens in the placing to actual tokens in the net (1). Then we add restrictions to Z according to the interval on the input arc of $\hat{t}$, and check if Z is consistent (2). In (3) we create a new existential zone Z', that equals Z after removing the tokens that are in the output places of the transition $\hat{t}$, and then adding to Z' the tightest constraint implied by either the input interval of $\hat{t}$ or the current constraints on the tokens, depending on which is tighter. This is how we get the interval that the age of token in the input place of $\hat{t}$ must be in s.t. the following theorem holds:

Theorem 6.4.
Given an existential zone Z, every zone $Z' \in pre_{\hat{t}}(Z)$ satisfies:

$$\forall M \models Z'.\ \exists M' \models Z \ s.t.\ M[\hat{t}\rangle M'$$

and $pre_{\hat{t}}(Z)$ is the largest such a set of zones.

Proof.
Given an arbitrary marking $M \models Z'$ we show that there exists a marking M' where $M[\hat{t}\rangle M'$, s.t. $M' \models Z$:

When we calculate $Z' = pre_{\hat{t}}(Z)$, the constraints in the DBM in Z' are the tightest constraints we can find by combining all the constraints in Z and the input interval $\mathcal{I}$. Thus, if $M \models Z'$, we ensure that when $\hat{t}$ fires s.t. $M[\hat{t}\rangle M'$, then $M' \models Z$ since all the constraints in Z are equally or more general than the constraints in Z'.

Given an arbitrary marking $M' \models Z$. we show that for any marking M where $M[\hat{t}\rangle M'$, $M \models Z'$ where $Z' \in pre_{\hat{t}}(Z)$:

All tokens created in the output places of $\hat{t}$ have to be of the same age. So the constraints on them, represented in Z, are combined to find one tightest constraint for which the age of each of these tokens fulfills. Furthermore, since the age of each of the tokens has to fit into the input interval $\mathcal{I}$ of $\hat{t}$ as well (the ages of the tokens in the output places are the same as the age of the token consumed by $\hat{t}$) we find one tightest constraint for both the constraints in Z and $\mathcal{I}$. This constraint is represented in the zone $Z' = pre_{\hat{t}}(Z)$ and any marking M, for which we can fire $\hat{t}$ and reach a marking $M' \models Z$, has to satisfy this zone Z'. □

Now the set $pre(Z)$, that contains all the zones in the age-preserving model, is defined as follows:

$$pre(Z) = \{pre_t(Z) | t \in T\} \cup \{pre_\epsilon(Z)\} \cup \{pre_{\hat{t}}(Z) | \hat{t} \in \hat{T}\}$$

Theorem 6.5 follows:

Theorem 6.5.
Coverability is decidable for age-preserving TAPNs in $\mathcal{N}_A(\mathbb{R}^+, Interv_3)$.

6.2.4 Summary

We have now shown how we can apply backwards reachability and existential zones to determine the decidability of coverability for TAPNs with age-preserving transitions.

6.3 Inhibitor Arcs

Inhibitor arcs for Petri nets were first defined in [16] but in this section we define them for TAPNs.

It is often the case, that we would like some transition to fire only if there are no tokens present in a given place. An example of this is when we are modeling mutual exclusion, where we want to fire a transition that enables a process to access a critical section if and only if no other process is accessing it. We model each process as an individual token in the net. In this case, we would like to be able to check if there are no tokens in the place "Critical Section". This is called *test for zero*, which is done by extending the TAPN model with *inhibitor arcs* [16]. Inhibitor arcs are graphically represented by a small circle at the origin of the arc. Figure 49 shows how a TAPN with inhibitor arcs can model mutual exclusion with two processes.

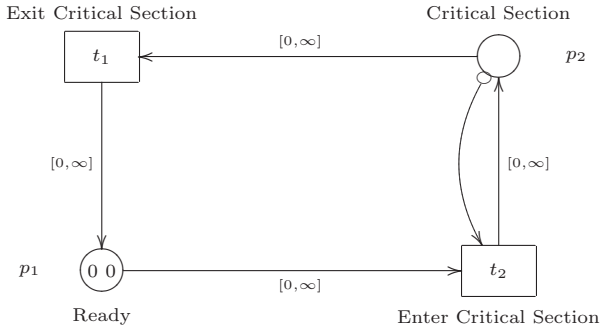


Figure 49: Mutual exclusion modeled with inhibitor arcs

6.3.1 TAPN with Inhibitor Arcs

Let us fix $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ and $Interv \in \{Interv_1, Interv_2, Interv_3\}$.

Definition 6.17.
*A **timed-arc Petri net (TAPN) with inhibitor arcs** is a 5-tuple* $N = (P, T, In, Out, Inhibit)$ *where*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places*
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions s.t.* $P \cap T = \emptyset$
- $In : P \times T \to Interv^{\oplus}$ *is a function that associates to each arc* (p, t) *a finite multiset of intervals*
- $Out : T \times P \to Interv^{\oplus}$ *is a function that associates to each arc* (t, p) *a finite multiset of intervals*
- $Inhibit : P \times T \to \{0, 1\}$ *is a function that represents the presence of an inhibitor arc from place to a transition*

We use the notation $^\circ t = \{p \mid Inhibit(p,t) = 1\}$ to denote the set of all input places of the transition t connected by an inhibitor arc.

Definition 6.18.
Let $N = (P, T, In, Out, Inhibit)$ be a TAPN with inhibitor arcs, M a marking on it and $t \in T$.
*We say that t is **enabled** at marking M if and only if:*

- $\forall p \in {}^\circ t.\ |M(p)| = 0$
- $\forall p \in {}^\bullet t.\ \exists X_p = \{x_1, \ldots, x_n\} \subseteq M(p)$, *where* $In(p,t) = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, *s.t.* $x_i \in \mathcal{I}_i$ *for* $1 \leq i \leq n$

i.e. all places connected to t by an inhibitor arc are empty and in each input place, for every regular input arc, there exists at least one token whose age fits the interval of that arc.

The set of tokens created, when t is fired, is any set

$$Y_p = \{y_1, \ldots, y_m\}\ \forall p \in t^\bullet, \text{ where } Out(t,p) = \{\mathcal{J}_1, \ldots, \mathcal{J}_m\}, \text{ s.t. } y_i \in \mathcal{J}_i \text{ for } 1 \leq i \leq m$$

i.e. in each output place, for every output arc, one token, whose age fits the interval of that arc, will be created.

Tokens are only consumed from input places of t and only created in output places of t. So we define:

$$X_p = \emptyset\ \forall p \in (P \setminus {}^\bullet t)$$
$$Y_p = \emptyset\ \forall p \in (P \setminus t^\bullet)$$

*Now we can define the **firing rule** formally.*

$$M'(p) = (M(p) \setminus X_p) \cup Y_p\ \forall p \in P$$

6.3.2 Discrete and Continuous Time

Theorem 6.6.
Timed-arc Petri nets with inhibitor arcs, using two or more inhibitor arcs, can not be reduced to regular timed-arc Petri nets.

Proof.
TAPNs (and even regular Petri nets) using inhibitor arcs are known to be Turing powerful [16]. If we were able to model a TAPN with inhibitor arcs using only regular arcs, then regular TAPNs were also Turing powerful, but that it is not the case, since coverability is decidable for them. □

6.3.3 Summary

We have defined TAPNs with inhibitor arcs and argued that this model is Turing powerful. We have shown how having inhibitor arcs in our model would give us a more power, since we could test for zero, but since it would make the model Turing powerful, all properties, e.g. coverability, would be undecidable.

6.4 Reset Arcs

In [10] the authors defined the reset Petri net model which is a regular Petri net model using reset arcs. In this section we extend that definition of reset arcs to work for TAPNs.

The difference between a reset arc and a regular arc, from place p to transition t, lies in the number of tokens consumed by t. Instead of consuming only one token, transition t consumes all tokens in p, thus *resetting* the token count in p to 0. Reset arcs are graphically represented with an x drawn over the arc, near its origin. Petri nets with reset arcs are called *reset Petri nets*.

We can also use reset arcs in timed-arc Petri nets. As for all other arcs in the TAPN model we need to assign time interval to each reset arc. Let us assume we have place p, transition t and a reset arc, with interval $\mathcal{I}$, leading from p to t. For t to become enabled (assuming p is its only input place) p has to have at least one token whose age fits this interval.

Let us look at an example where having reset arcs helps us. This example is an extension of the dairy example in Section 4.

> Each consumer that buys milk has a refrigerator where he can store the full milk bottles. When he consumes the milk, the empty bottle ends up in a recycle bin and the consumer is ready to buy a new, full one. At least 10 days after the first bottle is put in the recycle bin someone comes and takes all the bottles in it and returns them, all in one bottle case, back to the dairy.

Using TAPN with reset arc we can model this as is shown in Figure 50.

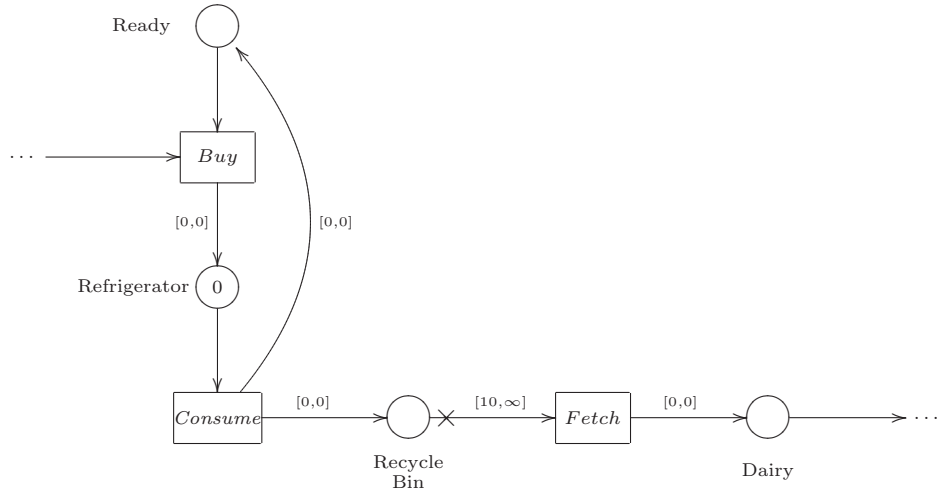


Figure 50: Reset arc example

A reset arc is used to model how all the empty milk bottles are taken from the recycle bin at once, as long as one of them has been in it for at least 10 days.

Let us now look at the formal definition.

6.4.1 TAPN with Reset Arcs

Let us fix $\mathcal{D} \in \{\mathbb{N}, \mathbb{R}^+\}$ and $Interv \in \{Interv_1, Interv_2, Interv_3\}$.

Definition 6.19.
*A **timed-arc Petri net (TAPN) with reset arcs** is a 5-tuple $N = (P, T, In, Out, Reset)$ where*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of places*
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of transitions s.t.* $P \cap T = \emptyset$
- $In : P \times T \to Interv^{\oplus}$ *is a function that associates to each arc* (p, t) *a finite multiset of intervals*
- $Out : T \times P \to Interv^{\oplus}$ *is a function that associates to each arc* (t, p) *a finite multiset of intervals*
- $Reset : P \times T \to Interv^{\oplus}$ *is a function that associates a time interval to each reset arc.*

We use the notation ${}^{\times}t = \{p \mid Reset(p, t) \neq \emptyset\}$ to denote the multiset of all input places of the transition t connected by a reset arc.

Definition 6.20.
Let $N = (P, T, In, Out, Reset)$ be a TAPN with reset arcs, M a marking on it and $t \in T$.
*We say that t is **enabled** at marking M if and only if:*

- $\forall p \in {}^{\bullet}t.\ \exists X_p = \{x_1, \ldots, x_n\} \subseteq M(p)$, *where* $In(p, t) = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, *s.t.* $x_i \in \mathcal{I}_i$ *for* $1 \leq i \leq n$
- $\forall p \in {}^{\times}t.\ \exists Z_p = \{z_1, \ldots, z_l\} \subseteq M(p)$, *where* $Reset(p, t) = \{\mathcal{K}_1, \ldots, \mathcal{K}_l\}$, *s.t.* $z_i \in \mathcal{K}_i$ *for* $1 \leq i \leq l$
- $X_p \cap Z_p = \emptyset$

i.e. in each input place, for every input arc, both regular and reset arcs, there exists at least one token whose age fits the interval of that arc.

The set of tokens created, when t is fired, is any set

$$Y_p = \{y_1, \ldots, y_m\}\ \forall p \in t^{\bullet}, \text{ where } Out(t, p) = \{\mathcal{J}_1, \ldots, \mathcal{J}_m\}, \text{ s.t. } y_i \in \mathcal{J}_i \text{ for } 1 \leq i \leq m$$

i.e. in each output place, for every output arc, one token, whose age fits the interval of that arc, will be created.

Tokens are only consumed from input places of t and only created in output places of t. So we define:

$$X_p = \emptyset \ \forall p \in (P \setminus {}^\bullet t)$$
$$Y_p = \emptyset \ \forall p \in (P \setminus t^\bullet)$$

And we define the ***firing rule*** *formally:*

$$M'(p) = \begin{cases} Y_p & \text{if } p \in {}^\times t \\ (M(p) \setminus X_p) \cup Y_p & \text{else} \end{cases} \quad \forall p \in P$$

So we remove a set of tokens, whose ages are within the intervals associated to the input arcs, from each input place of t, and add a set of new tokens to each output place of t, where the ages of the tokens are within the intervals associated to the output arcs. If the arc is a reset arc on the other hand, we remove all tokens from p and add only tokens to it, if it is an output place of t as well as an input place.

6.4.2 Discrete and Continuous Time

Theorem 6.7.
Timed-arc Petri nets with reset arcs, using two or more reset arcs, can not be reduced to regular timed-arc Petri nets.

Proof.
In [10] the authors show how to reduce nets with inhibitor arcs into nets with reset arcs, for regular Petri nets. We can extend this to TAPNs by simply associating each arc with a time interval $[0, \infty]$ and therefore not limiting the possible behavior of the net in any way. Being able to simulate inhibitor arcs in TAPNs with reset arcs tells us that this model is Turing powerful and thus that we can not reduce it to regular TAPNs since we know that coverability is decidable for them. □

6.4.3 Summary

We have defined TAPNs with reset arcs. Having reset arcs in our model can help us when modeling some systems but since we have proven this model to be Turing powerful, all its properties are undecidable.

7 Conclusion

In this paper we have looked at decidability and undecidability of some behavioral properties for both known and new Petri net models. We started with four properties in mind; reachability, coverability, liveness and deadlock but the fifth one, timed reachability, was taken into consideration when we found that reduction from continuous time DTAPN to PN, preserving timed-reachability, was an interesting thing to look at.

Decidability results for these five properties are listed in Tables 1-5. The following legend explains what each symbol represents:

- ✓= Shown by us to be decidable
- ✗= Shown by us to be undecidable
- (✓)= Shown by others to be decidable
- (✗)= Shown by others to be undecidable
- -= Not well defined
- ?= Open problem

Let us start by looking at the regular Petri net model (Table 1). Timed reachability is not a well defined property for this model since regular Petri nets are un-timed. Reachability was shown in [13] to be decidable and coverability was shown to be decidable in [16]. Reachability has been shown to be polynomially reducible to deadlock [6], and liveness polynomially reducible to liveness [6] making both of them decidable as well.

	Timed Reachability	Reachability	Coverability	Liveness	Deadlock
PN	–	(✓)	(✓)	(✓)	(✓)

Table 1: Summary of decidability results for PN

We classified the timed-arc Petri nets (Table 2) by both time domain and types of intervals. Timed reachability has been shown to be decidable for TAPNs with discrete time [8] but this remains an open problem for TAPNs with continuous time. Reachability is undecidable for all TAPNs [18], regardless of time domain and type of intervals. Coverability has been shown to be decidable for TAPNs using only closed intervals [3, 4] no matter the time domain. We were able to show that this also holds for TAPNs using both open and half open intervals, by modifying the algorithm used in [3, 4]. In Section 4 we gave polynomial reductions from reachability to deadlock and from deadlock to liveness for TAPNs

and since reachability is undecidable for all TAPNs, no matter the class, deadlock and liveness are undecidable as well.

	Timed Reachability	Reachability	Coverability	Liveness	Deadlock
TAPN					
$\mathcal{N}(\mathbb{N}, Interv_3)$	(✓)	(✗)	✓	✗	✗
$\mathcal{N}(\mathbb{N}, Interv_2)$	(✓)	(✗)	✓	✗	✗
$\mathcal{N}(\mathbb{N}, Interv_1)$	(✓)	(✗)	(✓)	✗	✗
$\mathcal{N}(\mathbb{R}^+, Interv_3)$	?	(✗)	✓	✗	✗
$\mathcal{N}(\mathbb{R}^+, Interv_2)$	?	(✗)	✓	✗	✗
$\mathcal{N}(\mathbb{R}^+, Interv_1)$	?	(✗)	(✓)	✗	✗

Table 2: Summary of decidability results for TAPN

Distributed timed-arc Petri nets (Table 3) were classified in the same way as regular TAPNs. We showed, with reduction to regular Petri nets, that timed reachability is decidable for DTAPNs with discrete time, but like for TAPNs this remains an open problem for DTAPNs with continuous time. Since reachability is undecidable for all TAPNs [18], reachability is undecidable for DTAPN as well, since it is a more expressive model. Coverability was claimed to be decidable for DTAPNs using only closed intervals [15] no matter the time domain but we give the proof for this. As for TAPNs, we were able to show that this also holds for DTAPNs using both open and half open intervals. Knowing that deadlock and liveness are undecidable for the less expressive TAPN model, it is obvious that these properties are undecidable for the DTAPN model as well.

	Timed Reachability	Reachability	Coverability	Liveness	Deadlock
DTAPN					
$\mathcal{N}_D(\mathbb{N}, Interv_3)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_D(\mathbb{N}, Interv_2)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_D(\mathbb{N}, Interv_1)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_D(\mathbb{R}^+, Interv_3)$	?	(✗)	✓	✗	✗
$\mathcal{N}_D(\mathbb{R}^+, Interv_2)$	?	(✗)	✓	✗	✗
$\mathcal{N}_D(\mathbb{R}^+, Interv_1)$	?	(✗)	✓	✗	✗

Table 3: Summary of decidability results for DTAPN

Age-preserving TAPN (Table 4) is a model which we defined ourselves in this paper so no decidable properties were known beforehand. We showed that the age-preserving TAPN model in discrete time is equivalent to the basic TAPN model and therefore that timed reachability is decidable for this model, in discrete time. But like for TAPNs and DTAPNs, this remains an open problem with continuous time. Since the age-preserving TAPN model is equivalent to the TAPN model in discrete time and more expressive in continuous time, it is obvious that reachability is undecidable for this model as well. By modifying the technique described in Section 5 in this paper we were able to show that coverability is decidable for all classes of age-preserving TAPNs. Deadlock and liveness are undecidable for this more expressive model.

	Timed Reachability	Reachability	Coverability	Liveness	Deadlock
Age-preserving TAPN					
$\mathcal{N}_A(\mathbb{N}, Interv_3)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_A(\mathbb{N}, Interv_2)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_A(\mathbb{N}, Interv_1)$	✓	(✗)	✓	✗	✗
$\mathcal{N}_A(\mathbb{R}^+, Interv_3)$	?	(✗)	✓	✗	✗
$\mathcal{N}_A(\mathbb{R}^+, Interv_2)$	?	(✗)	✓	✗	✗
$\mathcal{N}_A(\mathbb{R}^+, Interv_1)$	?	(✗)	✓	✗	✗

Table 4: Summary of decidability results for age-preserving TAPN

In this last table (Table 5) we have combined the results for both reset and inhibitor arcs, for both PN and TAPN. Timed reachability is, as mentioned in the text with Table 1, not well defined for the PN model. All other places in this Table 5 are marked as shown by others to be undecidable since inhibitor arcs have been proven to make PN, and thus TAPN, Turing powerful [16] and in [10] the authors gave a reduction from a PN using inhibitor arcs to a PN using reset arcs, thus showing that reset arcs are Turing powerful as well.

	Timed Reachability	Reachability	Coverability	Liveness	Deadlock
PN with inhibitor arcs	-	(✗)	(✗)	(✗)	(✗)
TAPN with inhibitor arcs	(✗)	(✗)	(✗)	(✗)	(✗)
PN with reset arcs	-	(✗)	(✗)	(✗)	(✗)
TAPN with reset arcs	(✗)	(✗)	(✗)	(✗)	(✗)

Table 5: Summary of decidability results for PN and TAPN using reset and inhibitor arcs

When looking at all five tables we see that only for timed-reachability do we not have results for every class, there are some open problems. The reason for this is that the idea to work on decidability results for other models than DTAPNs surfaced late in our project work, leaving insufficient time to explore all the interesting questions. Those "blanks" in the tables aside, we have listed complete decidability results for reachability, coverability, liveness and deadlock, for our six PN and TAPN models.

The general behavioral change that seems to occur when we extend the Petri net model with time is that reachability, deadlock and liveness all become undecidable, leaving coverability as the only decidable property. But with time we get a new property, timed reachability, which seems to be decidable for all our models in discrete time. For the continuous time models, timed reachability is an open problem but it would be interesting to know the results for them, especially to see if the use of open and half open intervals gives us more expressive models than those using only normal (closed) intervals.

References

[1] P. A. Abdulla, K. Cerans, B. Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, page 313. IEEE Computer Society, 1996.

[2] Parosh Aziz Abdulla. Personal email communication, 2005.

[3] Parosh Aziz Abdulla and Aletta Nylen. Timed Petri nets and BQOs. *Lecture Notes in Computer Science*, 2075:53–70, 2001.

[4] Parosh Aziz Abdulla and Aletta Nylen. Better quasi-ordered transition systems. *CoRR*, cs.LO/0409052, 2004.

[5] Fred D.J. Bowden. Modeling time in petri nets. In *Proceedings of the Second Australia-Japan Workshop on Stochastic Models*, 1996.

[6] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147:117–136, 1995.

[7] Allan Cheng. Reasoning about concurrent computational systems. Technical Report DS-96-2, August 1996. Ph.D. thesis. xiv+229 pp.

[8] David de Frutos Escrig, Valentín Valero Ruiz, and Olga Marroquín Alonso. Decidability of properties of timed-arc Petri nets. *Lecture Notes in Computer Science*, 1825:187+, 2000.

[9] Catherine Dufourd and Alain Finkel. Polynomial-time many-one reductions for Petri nets. In S. Ramesh and G. Sivakumar, editors, *Proceedings of the 17th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'97)*, volume 1346 of *Lecture Notes in Computer Science*, pages 312–326, Kharagpur, India, December 1997. Springer.

[10] Catherine Dufourd, Alain Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *ICALP '98: Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 103–115, London, UK, 1998. Springer-Verlag.

[11] J. Esparza and M. Nielsen. Decidability issues for petri nets - a survey. *Bulletin of the European Association for Theoretical Computer Science*, 52:245–262, 1994.

[12] M. Hack. Decidability questions for petri nets. Technical report, Cambridge, MA, USA, 1976.

[13] Ernst W. Mayr. An algorithm for the general petri net reachability problem. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 238–246. ACM Press, 1981.

[14] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Abril 1989.

[15] Mogens Nielsen, Vladimiro Sassone, and Jiri Srba. Towards a notion of distributed time for petri nets. In *Proceedings of the 22nd International Conference on Application and Theory of Petri Nets*, pages 23–31. Springer-Verlag, 2001.

[16] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, 1981.

[17] Olivier Roux, Didier Lime, and Guillaume Gardey. The softwares developed by the real-time systems group. `http://www.irccyn.ec-nantes.fr/irccyn/d/en/equipes/TempsReel/logs/software-2-romeo`, 2004.

[18] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On nondecidability of reachability for timed-arc petri nets. In *Proc. 8th Int. Workshop on Petri Net and Performance Models (PNPM'99), 8-10 October 1999, Zaragoza, Spain*, pages 188–196, 1999.

[19] J. Sifakis. Use of petri nets for performance evaulation. In *Proceedings of the Third International Symposium IFIP W.G. 7.3., Measuring, modelling and evaluating computer systems(Bonn-Bad Godesberg, 1977)*, pages 75–93. Elsevier Science Publishers, 1977.

[20] J. Srba. Timed-arc petri nets vs. networks of timed automata. In *Proceedings of 26nd International Conference on Application and Theory of Petri Nets (ICATPN 2005)*, LNCS. Springer-Verlag, 2005. To appear.

[21] Michael Weber. Petri net markup language. `http://www.informatik.hu-berlin.de/top/pnml/about.html`, 2004.