Bounded Model Checking Approaches for Verification of Distributed Time Petri Nets[⋆]

Artur Męski[1,2], Wojciech Penczek[2,3], Agata Półrola[1], Bożena Woźna-Szcześniak[4], and Andrzej Zbrzezny[4]

[1] University of Łódź, FMCS, Banacha 22, 90-238 Łódź, Poland
polrola@math.uni.lodz.pl
[2] Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, Poland
{meski,penczek}@ipipan.waw.pl
[3] University of Natural Sciences and Humanities, Institute of Informatics, 3 Maja 54, 08-110 Siedlce, Poland
[4] Jan Długosz University, IMCS, Armii Krajowej 13/15, 42-200 Częstochowa, Poland
{b.wozna,a.zbrzezny}@ajd.czest.pl

Abstract. We consider two symbolic approaches to bounded model checking (BMC) of distributed time Petri nets (DTPNs). We focus on the properties expressed in Linear Temporal Logic without the neXt-time operator ($\mathrm{LTL}_{-\mathrm{X}}$) and the existential fragment of Computation Tree Logic without the neXt-time operator ($\mathrm{ECTL}_{-\mathrm{X}}$). We give a translation of BMC to SAT and describe a BDD-based BMC for both $\mathrm{LTL}_{-\mathrm{X}}$ and $\mathrm{ECTL}_{-\mathrm{X}}$. The two translations have been implemented, tested, and compared with each other on two standard benchmarks. Our experimental results reveal the advantages and disadvantages of both the approaches.

1 Introduction

Verification of time dependent systems is a very active field of research. Many efficient approaches have been put forward for the verification of timed automata [1] and time Petri nets [22] by means of model checking [12, 26]. However, the *state explosion* still remains the main problem to deal with while verifying a timed system by searching through its state space, which in most cases is very large due to infinity of the dense time domain. Furthermore, the size of the state space is likely to grow exponentially in the number of the concurrent system components. Symbolic model checking techniques [21] can be used to overcome the above problem. These exploit various kinds of binary decision diagrams to represent the model [24] or are based on a translation to a propositional satisfiability problem.

Bounded model checking (BMC) is an efficient verification method using a model truncated up to a specific depth only. In turn, SAT-based BMC verification consists in translating a model checking problem solvable on a fraction of a

⋆ Partly supported by the Polish Ministry of Science and Higher Education under the grant No. N N206 258035.

model into a test of propositional satisfiability, which is then performed using a SAT-solver [28]. The method has been successfully applied to verification of both timed and untimed systems [3–5, 33]. Alternatively, one can use binary decision diagrams to represent a truncated model and to perform BDD-based verification [2, 13].

In this paper we investigate bounded model checking (BMC) approaches to verification of Distributed Time Petri Nets with discrete semantics, based on both SAT and BDD translations. There are several decisions taken that aim at making the verification of TPNs as efficient as possible. Below, we discuss them in detail to motivate clearly our approach. First of all, we believe that BMC is one of the main practical approaches, which can be used in case of dealing with huge or infinite state spaces. We motivate this point of view by comparing our experimental results with these of Tina, which operate on full state spaces. Clearly, BMC is restricted to verifying existential properties only, but it allows for tackling bounded models of large systems, whereas other approaches suffer from lack of memory.

Our second choice consists in dealing with distributed TPNs rather than with just 1-safe TPNs. The reason is that a representation of a global state contains only one clock for each process rather than one clock for each transition, which makes the encoding and the verification much more efficient. Another choice is related to the semantics. In this paper we investigate discrete semantics as we believe that in this case model checking is again more efficient. However, independently we are working on extending our approach to the dense semantics as we are aware that this is also a very interesting issue. Since there are several discrete semantics, we consider for each translation these which can be applied.

As to the temporal properties, we start with defining $\mathrm{CTL}^*_{-\mathrm{X}}$, but restrict ourselves two its two subsets $\mathrm{CTL}_{-\mathrm{X}}$ and $\mathrm{LTL}_{-\mathrm{X}}$, as these sublanguages allow for optimising the translations to SAT and BDDs. The languages do not contain the next step operator X as we are dealing with time systems, in which, for some discrete semantics, the next step may be not definable.

Next, we need to motivate our translations to SAT and BDDs. We are aware of the fact that there has been a tremendous speed-up due to applying the saturation technique [15] when performing decision diagram based verification. Moreover, the saturation combined with BMC was presented in [34], however only reachability checking was considered. Still, we believe, in most cases, BMC approach to BDD-based verification can be viewed as an alternative way of avoiding the BDD peak size when using BFS. In case of SAT we exploit the most efficient translations known for $\mathrm{ECTL}_{-\mathrm{X}}$ and $\mathrm{ELTL}_{-\mathrm{X}}$.

The above discussion motivates all the choices made in our paper and leads us to the main result, which is offering and comparing two symbolic BMC approaches for DTPNs. We show that for existential properties our BMC is much more efficient than Tina. We also compare efficiency of BMC depending on whether it is based on a translation to SAT or to BDD.

The main contribution of this paper is thus the combination of the three issues, as BMC has been studied, with both BDDs and, especially, SAT, but mostly for standard untimed models, while discrete time Petri nets have been studied with BDDs and extensions (e.g., [31]), but not for BMC.

To our best knowledge, no BMC method supporting ELTL$_{-\mathrm{X}}$ and ECTL$_{-\mathrm{X}}$ for time Petri nets has been defined so far, although some solutions for untimed Petri nets exist [27, 16]. Symbolic model checking has been investigated in many papers [2, 5]. Verification of CTL properties based on BDDs was introduced in [9]. In [27] SAT-based BMC for the existential fragment of CTL was described and implemented for elementary net systems. Verification methods using BDD-based BMC were studied in [10, 13] for simple invariant properties, in [23] for CTL over elementary nets systems, and in [19] for CTL extended with an epistemic component over multi-agent systems. On the other hand, verification of temporal properties for time Petri nets was a subject of intensive research of the teams of H. Boucheneb and O.H. Roux [6, 7, 20].

The rest of the paper is organised as follows. Section 2 presents logics LTL$_{-\mathrm{X}}$ and CTL$_{-\mathrm{X}}$. Section 3 introduces time Petri nets. SAT-based BMC for ELTL$_{-\mathrm{X}}$ and ECTL$_{-\mathrm{X}}$ is described in Section 4, whereas BDD-based BMC for these logics is in Section 5. Sections 6 and 7 contain respectively experimental results and concluding remarks.

2 Temporal Logics LTL$_{-\mathrm{X}}$ and CTL$_{-\mathrm{X}}$

We start with defining the logic CTL$^*_{-\mathrm{X}}$, which can express both linear- and branching-time properties. Then, we introduce variants of linear time temporal logic (LTL$_{-\mathrm{X}}$) as well as of branching time temporal logic (CTL$_{-\mathrm{X}}$) as sublogics of CTL$^*_{-\mathrm{X}}$. All the considered logics do not contain the next step operator X, which is reflected in their acronyms by $-$X.

Let PV be a set of propositional variables such that $\{true, false\} \subseteq PV$, and $\wp \in PV$. The language of CTL$^*_{-\mathrm{X}}$ is given as the set of all the state formulae φ_s (interpreted at states of a model), defined using path formulae φ_p (interpreted along paths of a model), by the following grammar:

$$\varphi_s := \wp \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \mathrm{A}\varphi_p$$
$$\varphi_p := \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \mathrm{U} \varphi_p \mid \varphi_p \mathrm{R} \varphi_p$$

In the above A ('for All paths') is a path quantifier, whereas U ('Until'), and R ('Release') are state operators. Further, the following standard abbreviations are used in writing CTL$^*_{-\mathrm{X}}$ formulae: $\varphi_s \wedge \varphi_s \overset{def}{=} \neg(\neg\varphi_s \vee \neg\varphi_s)$, $\varphi_p \wedge \varphi_p \overset{def}{=} \neg(\neg\varphi_p \vee \neg\varphi_p)$, $\mathrm{E}\varphi_p \overset{def}{=} \neg\mathrm{A}(\neg\varphi_p)$, $\mathrm{G}\varphi_p \overset{def}{=} false\mathrm{R}\varphi_p$, and $\mathrm{F}\varphi_p \overset{def}{=} true\mathrm{U}\varphi_p$.

Next, we define several sublogics of CTL$^*_{-\mathrm{X}}$ including variants of LTL$_{-\mathrm{X}}$ as well as of CTL$_{-\mathrm{X}}$. Although a standard model for LTL$_{-\mathrm{X}}$ is a path, for verification aims the logic is typically interpreted over all the paths of a Kripke model. So, two semantics are possible depending on whether a formula holds at all the paths or at some path of a model. Since we need to distinguish between these two semantics (in order to specify counterexamples), we find it useful to do it already at the level of the language by defining the universal (ALTL$_{-\mathrm{X}}$)

and the existential ($\mathrm{ELTL}_{-\mathrm{X}}$) versions of the logic. In the literature on the verification of linear time properties, if this distinction is not necessary, then $\mathrm{ALTL}_{-\mathrm{X}}$ is typically called $\mathrm{LTL}_{-\mathrm{X}}$.

$\mathrm{ALTL}_{-\mathrm{X}}$ ($\mathrm{ELTL}_{-\mathrm{X}}$) is the fragment of $\mathrm{CTL}^*_{-\mathrm{X}}$ in which only the formulae of the form $\mathrm{A}\varphi_p$ ($\mathrm{E}\varphi_p$, respectively) are allowed, where φ_p is a path formula which does not contain the path quantifiers A, E.

$\mathrm{CTL}_{-\mathrm{X}}$ is the fragment of $\mathrm{CTL}^*_{-\mathrm{X}}$ in which the syntax of path formulae is restricted such that each state operators must be preceded by a path quantifier (i.e., the modalities A, E, U, R can only appear paired in the combinations AU, EU, AR, ER).

$\mathrm{ACTL}_{-\mathrm{X}}$ ($\mathrm{ECTL}_{-\mathrm{X}}$) is the fragment of $\mathrm{CTL}_{-\mathrm{X}}$ such that the formulae are restricted to the positive boolean combinations of $\mathrm{A}(\varphi\mathrm{U}\psi)$ and $\mathrm{A}(\varphi\mathrm{R}\psi)$ ($\mathrm{E}(\varphi\mathrm{U}\psi)$ and $\mathrm{E}(\varphi\mathrm{R}\psi)$, respectively). Negation can be applied to propositions only.

A *model* for $\mathrm{CTL}^*_{-\mathrm{X}}$ is a Kripke structure $M = (L, S, s^0, \rightarrow, V)$, where L is a set of labels, S is a set of states, $s^0 \in S$ is the initial state, $\rightarrow\ \subseteq S \times L \times S$ is a total successor relation (i.e., $(\forall s \in S)(\exists s' \in S)(s \rightarrow s')$), and $V : S \longrightarrow 2^{PV}$ is a valuation function.

In our paper we assume the standard semantics of $\mathrm{CTL}^*_{-\mathrm{X}}$which can be found in several papers, among others in [11, 12], so we do not deliver it here. Moreover, we assume that a $\mathrm{CTL}^*_{-\mathrm{X}}$ formula φ is true in the model M (denoted by $M \models \varphi$) iff φ is true at the initial state of the model, i.e., $M, s^0 \models \varphi$.

3 Time Petri Nets

Let $\mathbb{N}$ denote the set of natural numbers. We start with a definition of time Petri nets:

Definition 1. *A time Petri net (TPN) is a tuple $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$, where $P = \{p_1, \ldots, p_{n_P}\}$ is a finite set of* places, *$T = \{t_1, \ldots, t_{n_T}\}$ is a finite set of* transitions, *$F \subseteq (P \times T) \cup (T \times P)$ is the* flow relation, *$m^0 \subseteq P$ is the* initial marking *of $\mathcal{N}$, and $Eft : T \rightarrow \mathbb{N}$, $Lft : T \rightarrow \mathbb{N} \cup \{\infty\}$ are functions describing the* earliest *and the* latest firing time *of the transition; where for each $t \in T$ we have $Eft(t) \leq Lft(t)$.*

For a transition $t \in T$ we define its *preset* $\bullet t = \{p \in P \mid (p, t) \in F\}$ and *postset* $t\bullet = \{p \in P \mid (t, p) \in F\}$, and consider only the nets such that for each transition the preset and the postset are nonempty. We need also the following notations and definitions:

– a *marking* of $\mathcal{N}$ is any subset $m \subseteq P$;
– a transition $t \in T$ is called *enabled* at m ($m[t\rangle$ for short) if $\bullet t \subseteq m$ and $t \bullet \cap (m \setminus \bullet t) = \emptyset$; and *leads from m to m'*, if it is enabled at m, and $m' = (m \setminus \bullet t) \cup t\bullet$. The marking m' is denoted by $m[t\rangle$ as well, if this does not lead to misunderstanding;
– $en(m) = \{t \in T \mid m[t\rangle\}$ is the set of all the transitions enabled at the marking m;

– a marking $m \subseteq P$ is *reachable* if there exists a sequence of transitions $t_1, \dots, t_l \in T$ and a sequence of markings $m_0, \dots, m_l$ such that $m_0 = m^0$, $m_l = m$, and for each $i \in \{1, \dots, l\}$ $t_i \in en(m_{i-1})$ and $m_i = m_{i-1}[t_i\rangle$;
– a marking m *concurrently enables* two transitions $t, t' \in T$ if $t \in en(m)$ and $t' \in en(m \setminus \bullet t)$;
– a net is *sequential* if no reachable marking of $\mathcal{N}$ concurrently enables two transitions.

It should be mentioned that the time Petri nets defined as above are often called *1-safe* in the literature. In this work we consider a subclass of TPNs – *distributed time Petri nets* (DTPNs) [26]:

Definition 2. *Let $\mathfrak{I} = \{\mathfrak{i}_1, \dots, \mathfrak{i}_n\}$ be a finite ordered set of indices, and let $\mathfrak{N} = \{N_\mathfrak{i} = (P_\mathfrak{i}, T_\mathfrak{i}, F_\mathfrak{i}, m^0_\mathfrak{i}, Eft_\mathfrak{i}, Lft_\mathfrak{i}) \mid \mathfrak{i} \in \mathfrak{I}\}$ be a family of 1-safe, sequential time Petri nets (called* processes*), indexed with $\mathfrak{I}$, with the pairwise disjoint sets $P_\mathfrak{i}$ of places, and satisfying the condition $(\forall \mathfrak{i}_1, \mathfrak{i}_2 \in \mathfrak{I})(\forall t \in T_{\mathfrak{i}_1} \cap T_{\mathfrak{i}_2})$ $(Eft_{\mathfrak{i}_1}(t) = Eft_{\mathfrak{i}_2}(t) \;\wedge\; Lft_{\mathfrak{i}_1}(t) = Lft_{\mathfrak{i}_2}(t))$. A* distributed time Petri net *$\mathcal{N} = (P, T, F, m^0, Eft, Lft)$ is the union of the processes $N_\mathfrak{i}$, i.e., $P = \bigcup_{\mathfrak{i} \in \mathfrak{I}} P_\mathfrak{i}$, $T = \bigcup_{\mathfrak{i} \in \mathfrak{I}} T_\mathfrak{i}$, $F = \bigcup_{\mathfrak{i} \in \mathfrak{I}} F_\mathfrak{i}$, $m^0 = \bigcup_{\mathfrak{i} \in \mathfrak{I}} m^0_\mathfrak{i}$, $Eft = \bigcup_{\mathfrak{i} \in \mathfrak{I}} Eft_\mathfrak{i}$, and $Lft = \bigcup_{\mathfrak{i} \in \mathfrak{I}} Lft_\mathfrak{i}$.*

Notice that the function $Eft_{\mathfrak{i}_1}$ ($Lft_{\mathfrak{i}_1}$) coincides with $Eft_{\mathfrak{i}_2}$ ($Lft_{\mathfrak{i}_2}$, resp.) for the joint transitions of each two processes $\mathfrak{i}_1$ and $\mathfrak{i}_2$. The interpretation of such a system is a collection of sequential, nondeterministic processes with communication capabilities (via joint transitions).

In what follows, we consider DTPNs only, assume that their initial markings contain exactly one place of each of the processes of the net, and that all their processes are *state machines* (i.e., for each $\mathfrak{i} \in \mathfrak{I}$ and each $t \in T_\mathfrak{i}$, it holds $|\bullet t| = |t \bullet| = 1$). This implies that in any marking of $\mathcal{N}$ there is exactly one place of each process. It is important to mention that a large class of distributed nets can be decomposed to satisfy the above requirement [18]. Moreover, for $t \in T$ we define $\mathcal{IV}(t) = \{\mathfrak{i} \in \mathfrak{I} \mid \bullet t \cap P_\mathfrak{i} \neq \emptyset\}$, and say that a process $N_\mathfrak{i}$ is *involved in a transition* t iff $\mathfrak{i} \in \mathcal{IV}(t)$.

3.1 Concrete State Spaces and Models

The current state of the net is given by its marking and the time passed since each of the enabled transitions became enabled (which influences the future behaviour of the net). In our work we assume a discrete-time semantics of DTPNs, i.e., consider integer time passings only (cf. [26]). Thus, a *concrete state* σ of a distributed TPN $\mathcal{N}$ can be defined as an ordered pair $(m, clock)$, where m is a marking, and $clock : \mathfrak{I} \to \mathbb{N}$ is a function which for each index $\mathfrak{i}$ of a process of $\mathcal{N}$ gives the time elapsed since the marked place of this process became marked most recently [29]. The set of all the concrete states is denoted by Σ. The initial state of $\mathcal{N}$ is $\sigma^0 = (m^0, clock^0)$, where m^0 is the initial marking, and $clock^0(\mathfrak{i}) = 0$ for each $\mathfrak{i} \in \mathfrak{I}$.

For $\delta \in \mathbb{N}$, let $clock + \delta$ denote the function given by $(clock + \delta)(\mathfrak{i}) = clock(\mathfrak{i}) + \delta$, and let $(m, clock) + \delta$ denote $(m, clock + \delta)$. The states of $\mathcal{N}$ can change when the time passes or a transition fires. In consequence, we introduce a labelled timed consecution relation $\rightarrow_c \subseteq \Sigma \times (T \cup \mathbb{N}) \times \Sigma$ given as follows:

– In a state $\sigma = (m, clock)$ a time $\delta \in \mathbb{N}$ can pass leading to a new state $\sigma' = (m, clock + \delta)$ (denoted $\sigma \xrightarrow{\delta}_c \sigma'$) iff for each $t \in en(m)$ there exists $\mathfrak{i} \in \mathcal{IV}(t)$ such that $clock(\mathfrak{i}) + \delta \leq Lft(t)$ (*time-successor relation*);
– In a state $\sigma = (m, clock)$ a transition $t \in T$ can fire leading to a new state $\sigma' = (m', clock')$ (denoted $\sigma \xrightarrow{t}_c \sigma'$) if $t \in en(m)$, for each $\mathfrak{i} \in \mathcal{IV}(t)$ we have $clock(\mathfrak{i}) \geq Eft(t)$, and there is $\mathfrak{i} \in \mathcal{IV}(t)$ such that $clock(\mathfrak{i}) \leq Lft(t)$. Then, $m' = m[t\rangle$,and for all $\mathfrak{i} \in \mathfrak{I}$ we have $clock'(\mathfrak{i}) = 0$ if $\mathfrak{i} \in \mathcal{IV}(t)$, and $clock'(\mathfrak{i}) = clock(\mathfrak{i})$ otherwise (*action-successor relation*).

Intuitively, the time-successor relation does not change the marking of the net, but increases the clocks of all the processes, provided that no enabled transition becomes disabled by passage of time (i.e., for each $t \in en(m)$ the clock of at least one process involved in the transition does not exceed $Lft(t)$). Firing of a transition t takes no time – the action-successor relation does not increase the clocks, but only sets to zero the clocks of the involved processes (note that each of these processes contains exactly one input and one output place of t, as the processes are state machines); and is allowed provided that t is enabled, the clocks of all the involved processes are greater than $Eft(t)$, and there is at least one such process whose clock does not exceed $Lft(t)$.

We define a *timed run* of $\mathcal{N}$ starting at a state $\sigma_0 \in \Sigma$ (σ_0-*run*) as a maximal sequence of concrete states, transitions, and time passings $\rho = \sigma_0 \xrightarrow{a_0}_c \sigma_1 \xrightarrow{a_1}_c \sigma_2 \xrightarrow{a_2}_c \ldots$, where $\sigma_i \in \Sigma$ and $a_i \in T \cup \mathbb{N}$ for all $i \in \mathbb{N}$. An *alternating run* is a timed run in which $a_i \in \mathbb{N}$ when i is even, and $a_i \in T$ when i is odd. A *non-alternating run* is a timed run with $a_i \in T \cup (\mathbb{N} \setminus \{0\})$ for all i. Given a set of propositional variables PV, we introduce a *valuation function* $V_c : \Sigma \rightarrow 2^{PV}$ which assigns the same propositions to the states with the same markings. We assume the set PV to be such that each $q \in PV$ corresponds to exactly one place $p \in P$, and use the same names for the propositions and the places. The function V_c is defined by $p \in V_c(\sigma) \Leftrightarrow p \in m$ for each $\sigma = (m, \cdot)$. The structure $M_c(\mathcal{N}) = (T \cup \mathbb{N}, \Sigma, \sigma^0, \rightarrow_c, V_c)$ is called a *concrete* (*discrete-timed*) *model of* $\mathcal{N}$.

3.2 A Model for $\mathbf{CTL^*_{-X}}$ Verification of DTPNs

The concrete model $M_c(\mathcal{N}) = (T \cup \mathbb{N}, \Sigma, \sigma^0, \rightarrow_c, V_c)$ for a DTPN $\mathcal{N}$ defined in Section 3 involves timed steps of arbitrary length. However, it can be proven that without loss of generality one can consider a model with a restricted set of timed labels, and of restricted values of the *clock* function. Let $c_{max}(\mathcal{N})$ denote the greatest finite value of Eft and Lft of the net $\mathcal{N}$, $\mathbf{c_{m1}}$ denote the value $c_{max}(\mathcal{N}) + 1$, and $\mathbb{C}_{\mathcal{N}}$ be the set of natural numbers from the interval $[0, \mathbf{c_{m1}}]$. Next, for a function $f : T \rightarrow \mathbb{N}$ and $a \in \mathbb{N}$, let $f|_a$ denote the function given by $f|_a(t) = f(t)$ if $f(t) \leq a$, and $f|_a(t) = a$ otherwise. Let $clock_s : \mathfrak{I} \rightarrow \mathbb{C}_{\mathcal{N}}$ denote the function which for each index $\mathfrak{i}$ of a process of $\mathcal{N}$ gives the time either elapsed since the marked place of this process became marked most recently, or "frozen" on the value $\mathbf{c_{m1}}$ if the time elapsed since the marked place becomes marked exceeded $c_{max}(\mathcal{N})$. Let $\sigma|_{\mathbf{c_{m1}}}$, for $\sigma = (m, clock) \in \Sigma$, be the state $(m, clock_s)$ with $clock_s = clock|_{\mathbf{c_{m1}}}$. Moreover, for $\delta \in \mathbb{N}$, let $clock_s \oplus \delta$ denote the function given by $(clock_s \oplus \delta)(\mathtt{i}) = clock_s(\mathtt{i}) + \delta$ if $clock_s(\mathtt{i}) + \delta \leq \mathbf{c_{m1}}$, and $(clock_s \oplus \delta)(\mathtt{i}) = \mathbf{c_{m1}}$ otherwise. The *reduced (discrete-timed) model* for DTPN $\mathcal{N}$ is defined as follows: $\widehat{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma_s, \sigma^0, \rightarrow_s, V_s)$, where $\Sigma_s = \{\sigma|_{\mathbf{c_{m1}}} \mid \sigma \in \Sigma\}$, V_s is given by $V_s(\sigma|_{\mathbf{c_{m1}}}) = V_c(\sigma)$, and the relation $\rightarrow_s \subseteq \Sigma_s \times (T \cup \mathbb{C}_{\mathcal{N}}) \times \Sigma_s$ is defined by

- in a state $\sigma_s = (m, clock_s)$ a time $\delta \in \mathbb{C}_{\mathcal{N}}$ can pass leading to a new state $\sigma'_s = (m, clock_s \oplus \delta)$ (denoted $\sigma_s \xrightarrow{\delta}_s \sigma'_s$) iff for each $t \in en(m)$ there exists $\mathtt{i} \in \mathcal{IV}(t)$ such that $clock_s(\mathtt{i}) \oplus \delta \leq Lft(t)$,
- a transition $t \in T$ can fire in a state $\sigma_s = \sigma|_{\mathbf{c_{m1}}}$ leading to a state σ'_s (denoted $\sigma_s \xrightarrow{t}_s \sigma'_s$) iff $\sigma \xrightarrow{t}_c \sigma'$ for some $\sigma' \in \Sigma$ s.t. $\sigma'_s = \sigma'|_{\mathbf{c_{m1}}}$.

In order to show that $\widehat{M}_c(\mathcal{N})$ can replace $M_c(\mathcal{N})$ in CTL$^*_{-\mathrm{X}}$ verification we shall prove the following lemma:

Lemma 1. *For a given DTPN $\mathcal{N}$ the models $M_c(\mathcal{N}) = (T \cup \mathbb{N}, \Sigma, \sigma^0, \rightarrow_c, V_c)$ and $\widehat{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma_s, \sigma^0, \rightarrow_s, V_s)$ are bisimulation equivalent.*

The proof can be found in the appendix. In the proof we use an "intermediate" model $\widetilde{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma, \sigma^0, \rightarrow_r, V_c)$ with $\rightarrow_r \subseteq \Sigma \times (T \cup \mathbb{C}_{\mathcal{N}}) \times \Sigma$ given by

- in a state $\sigma = (m, clock)$ a time $\delta \in \mathbb{C}_{\mathcal{N}}$ can pass leading to a new state $\sigma' = (m, clock + \delta)$ (denoted $\sigma \xrightarrow{\delta}_r \sigma'$) iff for each $t \in en(m)$ there exists $\mathtt{i} \in \mathcal{IV}(t)$ such that $clock(\mathtt{i}) + \delta \leq Lft(t)$,
- a transition $t \in T$ can fire in a state σ leading to a state σ' ($\sigma \xrightarrow{t}_r \sigma'$) iff $\sigma \xrightarrow{t}_c \sigma'$,

(i.e., the model which differs from $\widehat{M}_c(\mathcal{N})$ in such a way that the values of the *clock* function are not restricted to $\mathbf{c_{m1}}$) which is bisimulation equivalent to $M_c(\mathcal{N})$. Further, we define the following equivalence relation, which is used in the next section to define a SAT-based BMC method.

Definition 3. *Let $\sigma = (m, clock)$ and $\sigma' = (m', clock')$ be two states of a DTPN $\mathcal{N}$ ($\sigma, \sigma' \in \Sigma$). The states σ, σ' are* $\star$-equivalent *(denoted $\sigma \simeq_\star \sigma'$) iff $m = m'$ and $\forall_{t \in en(m)}[(\min_{\mathtt{i} \in \mathcal{IV}(t)} clock(i) = \min_{i \in \mathcal{IV}(t)} clock'(i) \wedge \min_{i \in \mathcal{IV}(t)} clock(\mathtt{i}) \leq c_{max}(\mathcal{N})) \vee (\min_{\mathtt{i} \in \mathcal{IV}(t)} clock(i) > c_{max}(\mathcal{N}) \wedge \min_{\mathtt{i} \in \mathcal{IV}(t)} clock'(i) > c_{max}(\mathcal{N}))]$.*

The following lemma shows that the equivalence preserves the behaviours of the net. Its proof can be found in the appendix.

Lemma 2. *Let $\sigma, \sigma' \in \Sigma$ be $\star$-equivalent. Thus, for any $l \in T \cup \mathbb{N}$ we have:*

- *if $\sigma \xrightarrow{l}_c \sigma_1$ for some $\sigma_1 \in \Sigma$ then there is $\sigma'_1 \in \Sigma$ s.t. $\sigma' \xrightarrow{l}_c \sigma'_1$ and $\sigma_1 \simeq_\star \sigma'_1$,*
- *if $\sigma' \xrightarrow{l}_c \sigma'_1$ for some $\sigma'_1 \in \Sigma$ then there is $\sigma_1 \in \Sigma$ s.t. $\sigma \xrightarrow{l}_c \sigma_1$ and $\sigma'_1 \simeq_\star \sigma_1$.*

4 SAT-Based BMC for ELTL$_{-\mathrm{X}}$ and ECTL$_{-\mathrm{X}}$

BMC is a verification technique whose main idea consists in considering a model truncated up to a specific depth. Thus, BMC is mostly used to find counterexamples for the properties expressed in "universal" logics (in our case ACTL$_{-\mathrm{X}}$ and

ALTL$_{-\mathrm{X}}$), or to prove that properties expressed in "existential" logics (ECTL$_{-\mathrm{X}}$, ELTL$_{-\mathrm{X}}$) hold.

The BMC method used in our paper is based on the BMC method for the existential fragment of CTL$^*_{-\mathrm{X}}$ (ECTL$^*_{-\mathrm{X}}$) [32], and on an improved BMC translation for the ECTL$_{-\mathrm{X}}$ fragment [35]. In particular, in the paper we adapt the BMC techniques mentioned above to the DTPN setting. Let $\widetilde{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma, \sigma^0, \rightarrow_r, V_c)$ be a model for a given DTPN $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$, and φ an ECTL$_{-\mathrm{X}}$ or ELTL$_{-\mathrm{X}}$ formula describing an undesired property. To show that φ is true in $\widetilde{M}_c(\mathcal{N})$, it is enough to prove that φ holds in a fragment (submodel) M' of $\widetilde{M}$. Thus, we start by taking a submodel M' of the model $\widetilde{M}_c(\mathcal{N})$ that consists of the finite prefixes of paths from $\widetilde{M}_c(\mathcal{N})$ restricted by a bound $k \in \mathbb{N}$ – traditionally called k-paths. The number of k-paths in M' depends on the checked formula φ, and it is specified by a value of a function $f_k : \mathcal{F}_{\mathrm{ECTL}^*_{-\mathrm{X}}} \rightarrow \mathbb{N}$ defined by: for $\wp \in PV$, $f_k(\wp) = f_k(\neg\wp) = 0$, $f_k(\varphi \wedge \psi) = f_k(\varphi) + f_k(\psi)$, $f_k(\varphi \vee \psi) = \max\{f_k(\varphi), f_k(\psi)\}$, $f_k(E\varphi) = f_k(\varphi) + 1$, $f_k(\varphi \mathrm{U} \psi) = k \cdot f_k(\varphi) + f_k(\psi)$, $f_k(\varphi \mathrm{R} \psi) = (k+1) \cdot f_k(\psi) + f_k(\varphi)$. Next, we translate the problem of checking whether the M' is a model for φ to the problem of checking the satisfiability of the following propositional formula:

$$[\widetilde{M}_c(\mathcal{N}), \varphi]_k := [\widetilde{M}_c(\mathcal{N})^{\varphi, \sigma^0}]_k \wedge [\varphi]_{M'} \tag{1}$$

The first conjunct of Formula 1 represents all the submodels M' of $\widetilde{M}_c(\mathcal{N})$ that consists of $f_k(\varphi)$ k-paths, and the second a number of constraints that must be satisfied on these submodels for φ to be satisfied. Once this translation is defined, satisfiability of an ECTL$_{-\mathrm{X}}$ or ELTL$_{-\mathrm{X}}$ formula can be tested with a SAT-solver.

How to define the formula $[\widetilde{M}_c(\mathcal{N})^{\varphi, \sigma^0}]_k$ in the DTPN settings we show in the next subsection. Note however that for a given DTPN $\mathcal{N}$, the formula $[\widetilde{M}_c(\mathcal{N})^{\varphi, \sigma^0}]_k$ can be implemented either using the model $\widetilde{M}_c(\mathcal{N})$ or using $\widehat{M}_c(\mathcal{N})$. We have chosen $\widetilde{M}_c(\mathcal{N})$ in order to simplify the implementation. It should be explained that although in $\widetilde{M}_c(\mathcal{N})$ there is no upper bound on the values of clocks, restricting the lengths of the time steps allows to bound the values of clocks on k-paths by a value depending on k and $c_{max}(\mathcal{N})$. The definition of the formula $[\varphi]_{M'}$ depends on whether φ is in ECTL$_{-\mathrm{X}}$ or in ELTL$_{-\mathrm{X}}$, and whether considered k-paths are, or are not *loops*; a k-path $\pi_k = (\sigma_0, \sigma_1, \ldots, \sigma_k)$ is called a (k,l)-*loop*, if

– $\sigma_k \simeq_\star \sigma_l$ for some $0 \leq l < k$ (the non-alternating semantics).
– $\sigma_k \simeq_\star \sigma_l$ for some $0 \leq l < k$, and either both k and l are odd or they are both even (the alternating semantics).

The difference in the above definitions follows from the fact that in the alternating semantics the looping runs need to preserve the alternating structure when "unfolded", while in the non-alternating semantics their stucture is preserved without any additional conditions. Using $\simeq_\star$ instead of the standard equality of states follows from the fact that for the further possible behaviours of the net at a given state only the minimal values of the clocks of the processes involved in the enabled transitions are important.

A k-path π_k is called a *loop*, if it is (k,l)-loop for some $l \in \{0,\dots,k-1\}$. In this paper we assume the definitions of $[\varphi]_{M'}$ that can be found, respectively, in [35] (ECTL$_{-\mathrm{X}}$), and in [32, 5] (ELTL$_{-\mathrm{X}}$). However, to apply them to the DTPN setting, we have changed the definition of the loop to the one presented above.

Definition of formula $[\widetilde{M}_c(\mathcal{N})^{\varphi,\sigma^0}]_k$. Let $\widetilde{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma, \sigma^0, \to_r, V_c)$ be a model of a given DTPN $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$, φ an ECTL$_{-\mathrm{X}}$ or ELTL$_{-\mathrm{X}}$ formula, and $k \in \mathbb{N}$ a bound. In order to define the formula $[\widetilde{M}_c(\mathcal{N})^{\varphi,\sigma^0}]_k$ that constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-paths of $\widetilde{M}_c(\mathcal{N})$, we proceed as follows. We assume that each state $\sigma \in \Sigma$ is given in a unique binary form, i.e., every state σ can be encoded as a bit vector $(\sigma[1],\dots,\sigma[\mathfrak{l}_b])$ of length $\mathfrak{l}_b$ depending on the number of places P of $\mathcal{N}$, the bound k , the value $c_{max}(\mathcal{N})$ (i.e., the greatest finite value of Eft and Lft), and the value $f_k(\varphi)$. Thus, each state σ can be represented by a valuation of a vector $w = (w[1],\dots,w[\mathfrak{l}_b])$ (called a *global state variable*), where $w[i]$, for $i = 1,\dots,\mathfrak{l}_b$ is a propositional variable (called *state variable*)[5]. A finite sequence $(w_0,\dots,w_k)$ of global state variables is called a *symbolic k-path*. Since in the ECTL$_{-\mathrm{X}}$ case we shall need to consider not just one but a number of symbolic k-paths, we use the notation of j-th symbolic k-path $(w_{0,j},\dots,w_{k,j})$, where $w_{i,j}$ are global state variables for $0 \le j < f_k(\varphi)$ and $0 \le i \le k$; the number of symbolic k-paths depends on the formula φ under investigation, and it is returned as the value $f_k(\varphi)$ of the function f_k; note that if φ is an ELTL$_{-\mathrm{X}}$ formulae then $f_k(\varphi) = 1$.

Let w, w' be two global state variables. We define the following auxiliary propositional formulae:

- $I_\sigma(w)$ is a formula that encodes the state σ of the model $\widetilde{M}_c(\mathcal{N})$, i.e., $\sigma[i] = 1$ is encoded by $w[i]$, and $\sigma[i] = 0$ is encoded by $\neg w[i]$.
- $\mathcal{TS}(w,w')$ ($\mathcal{TS}'(w,w')$) is a formula over w and w' which is true for two valuations σ_w of w and $\sigma_{w'}$ of w' iff $\sigma_w \xrightarrow{\delta}_r \sigma_{w'}$, for $\delta \in \mathbb{C}_{\mathcal{N}}$ ($\delta \in \mathbb{C}_{\mathcal{N}} \setminus \{0\}$, respectively). It encodes the time-successor relation of $\widetilde{M}_c(\mathcal{N})$.
- $\mathcal{AS}(w,w')$ is a formula over w and w' which is true for two valuations σ_w of w and $\sigma_{w'}$ of w' iff $\sigma_w \xrightarrow{t}_r \sigma_{w'}$, for $t \in T$. It encodes the action-successor relation of $\widetilde{M}_c(\mathcal{N})$.

The propositional formula $[\widetilde{M}_c^{\varphi,\sigma^0}]_k$ is defined as follows:

$$[\widetilde{M}_c^{\varphi,\sigma^0}]_k := I_{\sigma^0}(w_{0,0}) \wedge \bigwedge_{j=0}^{f_k(\varphi)-1} \bigwedge_{i=0}^{k-1} \mathcal{R}(w_{i,j}, w_{i+1,j})$$

where $w_{i,j}$ for $0 \le i \le k$ and $0 \le j < f_k(\varphi)$ are global state variables, and

(a) $\mathcal{R}(w_{i,j}, w_{i+1,j}) := \mathcal{TS}(w_{i,j}, w_{i+1,j})$ when i is even, and $\mathcal{R}(w_{i,j}, w_{i+1,j}) := \mathcal{AS}(w_{i,j}, w_{i+1,j})$ when i is odd (the alternating semantics), or

(b) $\mathcal{R}(w_{i,j}, w_{i+1,j}) := \mathcal{TS}'(w_{i,j}, w_{i+1,j}) \vee \mathcal{AS}(w_{i,j}, w_{i+1,j})$ (the non-alternating semantics).

[5] Notice that we distinguish between states of Σ encoded as sequences of 0's and 1's (we refer to these as valuations of w), and their representations in terms of propositional variables $w[i]$.

Note that if φ is an ELTL$_{-\mathrm{X}}$ formula, then $f_k(\varphi) = 1$, and the above definition is equivalent to the following one: $[\widetilde{M}_c^{\varphi,\sigma^0}]_k := I_{\sigma^0}(w_{0,0}) \wedge \bigwedge_{i=0}^{k-1} \mathcal{R}(w_{i,0}, w_{i+1,0})$.

5 BDD-based BMC for ELTL$_{-\mathrm{X}}$ and ECTL$_{-\mathrm{X}}$

Binary decision diagrams (BDDs) [8, 17] are an efficient data structure widely used for storing and manipulating boolean functions. In the paper we use Reduced Ordered Binary Decision Diagrams (ROBDDs) instead of the "pure" BDD structures. The advantage of ROBDDs is that they are canonical for a particular function and variable order.

To introduce a BDD-based bounded model checking method, we start with describing ECTL$_{-\mathrm{X}}$ in terms of sets of reachable states at which the given formula holds [17]. For this purpose we need the notion of a fixed point.

Let S be a finite set and $\tau : 2^S \longrightarrow 2^S$ a *monotone* function, i.e., $X \subseteq Y$ implies $\tau(X) \subseteq \tau(Y)$ for all $X, Y \subseteq S$. Let $\tau^i(X)$ be defined by $\tau^0(X) = X$ and $\tau^{i+1}(X) = \tau(\tau^i(X))$. We say that $X' \subseteq S$ is a *fixed point* of τ if $\tau(X') = X'$. It can be proven that if τ is monotone, S is a finite set and $|S|$ is a number of its elements, then there exist $m, n \leq |S|$ such that $\tau^m(\emptyset)$ is the least fixed point of τ (denoted by $\mu X.\tau(X)$) and $\tau^n(S)$ is the greatest fixed point of τ (denoted by $\nu X.\tau(X)$).

Let $M = (L, S, s^0, \rightarrow, V)$ be a model, and $S_R \subseteq S$ a set of all the reachable states of the model M. For $X \subseteq S_R$, let $pre_\exists(X) = \{s \in S_R \mid (\exists s' \in X)(\exists l \in L)\ s \xrightarrow{l} s'\}$ be a set of all the reachable states from which there is a transition to some state in X. Further, we denote the set of all the reachable states of the model M at which φ holds by $[\![M, \varphi]\!]$ or by $[\![\varphi]\!]$, if M is implicitly understood. For ECTL$_{-\mathrm{X}}$ formulae φ and ψ we define the following sets: $[\![true]\!] \overset{def}{=} S_R$, $[\![\wp]\!] \overset{def}{=} \{s \in S_R \mid \wp \in V(s)\}$, $[\![\neg\varphi]\!] \overset{def}{=} S_R \setminus [\![\varphi]\!]$, $[\![\varphi \wedge \psi]\!] \overset{def}{=} [\![\varphi]\!] \cap [\![\psi]\!]$, $[\![\varphi \vee \psi]\!] \overset{def}{=} [\![\varphi]\!] \cup [\![\psi]\!]$. The remaining operators can be defined as fixed points in the following way: $[\![\mathrm{EG}\varphi]\!] \overset{def}{=} \nu X.[\![\varphi]\!] \cap pre_\exists(X)$, $[\![\mathrm{E}[\varphi \mathrm{U} \psi]]\!] \overset{def}{=} \mu X.[\![\psi]\!] \cup ([\![\varphi]\!] \cap pre_\exists(X))$.

To define the sets corresponding to ELTL$_{-\mathrm{X}}$ formulae we proceed as follows. Let $M = (L, S, s^0, \rightarrow, V)$ be a model, and φ an ELTL$_{-\mathrm{X}}$ formula. We begin with constructing the tableau for φ, as described in [11], that is then combined with the model M to obtain their product, which contains these paths of M where the formula φ potentially holds. The product is then verified in terms of CTL model checking of EG$true$ formula under fairness constraints. The fairness constraints, corresponding to sets of states, allow to choose only these paths of the model, along which at least one state in each set representing fairness constraints appears infinitely often. In the case of ELTL$_{-\mathrm{X}}$ model checking, fairness is applied to guarantee that $\mathrm{E}(\varphi \mathrm{U} \psi)$ really holds, i.e., to eliminate paths where φ holds continuously, but ψ never holds. Finally, we choose only these reachable states of the product that belong to some particular set of states computed for the formula. The corresponding states of the verified system that are in this set, comprise the set $[\![M, \varphi]\!]$, i.e., the set of the reachable states where the verified formula holds. As we are unable to include a more detailed description of the method (due to the page limit), we refer the reader to [11] for more details.

Before describing the BDD-based bounded model checking method, we first define a submodel. Namely, for the model $M = (L, S, s^0, \rightarrow, V)$ and $U \subseteq S$ such that $s^0 \in U$, we define a *submodel* $M|_U = (L', U, s^0, \rightarrow', V')$, where: $L' = \{l \in L \mid (\exists s, s' \in U)\ s \xrightarrow{l} s'\}$, $\rightarrow' = \{s \xrightarrow{l} s' \mid s, s' \in U\}$, $V' : U \longrightarrow 2^{PV}$ is defined by $V'(s) = V(s)$ for all $s \in U$. As the method can be applied to BMC of both ECTL$_{-\mathrm{X}}$ and ELTL$_{-\mathrm{X}}$, we do not distinguish between ECTL$_{-\mathrm{X}}$ and ELTL$_{-\mathrm{X}}$ formulae, and in what follows, by φ we understand either an ECTL$_{-\mathrm{X}}$ formula or an ELTL$_{-\mathrm{X}}$ formula.

Let $M = (L, S, s^0, \rightarrow, V)$ be a model. For any set $X \subseteq S$ we define the set of successors of all the states in X by $X_{\leadsto} \overset{def}{=} \{s' \in S \mid (\exists s \in X)(\exists l \in L)\ s \xrightarrow{l} s'\}$. The complete set of the reachable states is obtained by computing the least fixed point $\mu Reach.\{s^0\} \cup Reach \cup Reach_{\leadsto}$. With each iteration, when the set *Reach* is extended with new states, i.e., with the set $Reach_{\leadsto}$, the verified formula is checked in the submodel $M|_{Reach}$. The loop terminates and the algorithm returns *true*, if the initial state s^0 is in the set of states of the obtained submodel at which φ holds. The search continues until no new states can be discovered from the states in *Reach*, i.e., the fixed point is reached. When we obtain the complete set of reachable states, and a path from the initial state on which φ holds could not be found in any of the obtained submodels, the algorithm terminates with *false*.

BDD-based Verification of DTPNs In order to verify a DTPN using BDDs first we need to translate its underlying reduced model into boolean formulae that are encoded with BDDs. Let $\widehat{M_c}(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma_s, \sigma^0, \rightarrow_s, V_s)$ be a model of a given DTPN $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$. We assume that every state $\sigma \in \Sigma_s$ can be encoded as a bit vector $(\sigma[1], \ldots, \sigma[\mathfrak{l}_b])$ of length $\mathfrak{l}_b$ depending on the number of places P of $\mathcal{N}$, and the value $c_{max}(\mathcal{N})$. Thus, each state σ can be represented by a valuation of a vector $w = (w[1], \ldots, w[\mathfrak{l}_b])$ (called a *global state variable*), where $w[i]$, for $i = 1, \ldots, \mathfrak{l}_b$ is a propositional variable (called *state variable*).

Let w, w' be two global state variables. We define the following boolean formulae that are used in the encoding:

- $\mathcal{I}_\sigma(w)$ is a formula that encodes the state σ of the model $\widehat{M}(\mathcal{N})$, i.e., $\sigma[i] = 1$ is encoded by $w[i]$, and $\sigma[i] = 0$ is encoded by $\neg w[i]$.
- $TS(w, w')$ is a formula over w and w' which is true for two valuations σ_w of w and $\sigma_{w'}$ of w' iff $\sigma_w \xrightarrow{\delta}_s \sigma_{w'}$, for $\delta \in \mathbb{C}_{\mathcal{N}} \setminus \{0\}$. It encodes the time-successor relation of $\widehat{M_c}(\mathcal{N})$.
- $AS_t(w, w')$, where $t \in T$, is a formula over w and w' which is true for two valuations σ_w of w and $\sigma_{w'}$ of w' iff $\sigma_w \xrightarrow{t}_s \sigma_{w'}$. It encodes the action-successor relation of $\widehat{M_c}(\mathcal{N})$ for the transition $t \in T$.

- $T(w, w') = \left(\bigvee_{t \in T} AS_t(w, w')\right) \vee TS(w, w')$ is a formula over w and w' which is true for two valuations σ_w of w and $\sigma_{w'}$ of w' iff $\sigma_w \rightarrow_s \sigma_{w'}$. It encodes the transition relation of $\widehat{M_c}(\mathcal{N})$.

Notice that due to the fact that an implementation of the alternating semantics would be inefficient in the case of the BDD-based method, we apply only the non-alternating semantics.

In our implementation we use the order of variables suggested in [17] where the variables encoding the states and their successors are interleaved. The explanation of how we can compute the BDDs for the sets $X_{\leadsto}$ and $pre_{\exists}(X)$ (where $X \in \Sigma_s$) that are needed by the described fixed point methods can be found also in [17]. Moreover, we encode each disjunct of the formula encoding the transition relation, with separate BDDs.

6 Experimental Results

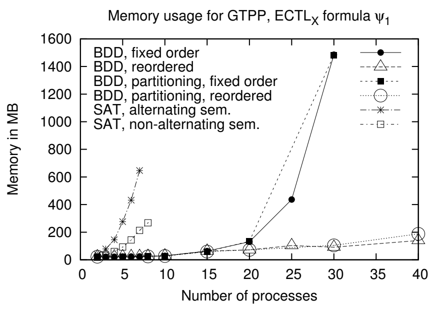
In this section we consider two scalable DTPNs which we use to evaluate the performance of our SAT- and BDD-based BMC algorithms, as well as of the tool Tina, for the verification of several properties expressed in $\mathrm{ECTL}_{-\mathrm{X}}$ and $\mathrm{ELTL}_{-\mathrm{X}}$. The evaluation is given by means of the running time and the consumed memory. Graphs representing the benchmarks described below can be found at the webpage of VerICS – http://verics.ipipan.waw.pl/.

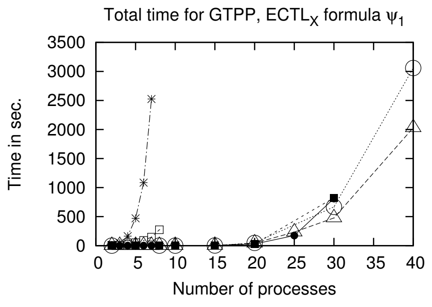
The first benchmark we consider is the Generic Timed Pipeline Paradigm (GTPP) Petri net model [25], which consists of Producer producing data (*ProdReady*) or being inactive, Consumer receiving data (*ConsReady*) or being inactive, and a chain of n intermediate Nodes which can be ready for receiving data ($Node_iReady$), processing data ($Node_iProc$), or sending data ($Node_iSend$). The example can be scaled in the number of intermediate nodes. The intervals are used to adjust the time properties of Producer, Consumer, and of the intermediate Nodes.

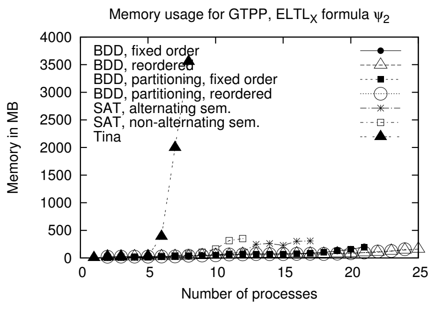
The second benchmark of our interest is the DTPN model for *Fischer's mutual exclusion protocol* (Mutex). The model consists of n time Petri nets, each one modelling a process, plus one additional net used to coordinate the access of processes to their critical sections *Mutual exclusion* means that no two processes are in their critical sections at the same time. The preservation of this property depends on the relative values of the time-delay constants δ and Δ. In particular, Fischer's protocol ensures mutual exclusion iff $\Delta < \delta$. This DTPN can be scaled in the number of processes.

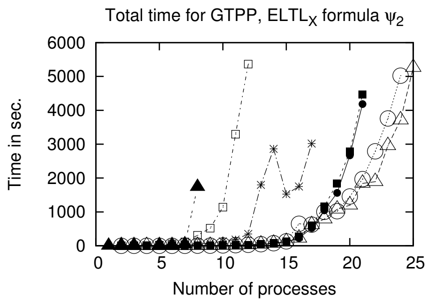
The GTPP Petri net model, where all the intervals are set to $[0, 2]$, was tested with the $\mathrm{ECTL}_{-\mathrm{X}}$ formula $\psi_1 = \mathrm{EG}(\mathrm{EF}(\neg ConsReady))$, and the $\mathrm{ELTL}_{-\mathrm{X}}$ formula $\psi_2 = \mathrm{EGF}(\neg ConsReady)$. The Mutex protocol, with $\Delta = 1$ and $\delta = 2$, was tested with the $\mathrm{ECTL}_{-\mathrm{X}}$ formulae: $\psi_1 = \mathrm{EGEF}(critical1 \vee \ldots \vee criticalN)$, $\psi_2 = \mathrm{EF}(trying_1 \wedge \ldots \wedge trying_N \wedge \mathrm{EG}(\neg critical_2 \wedge \ldots \wedge \neg critical_N))$, and the $\mathrm{ELTL}_{-\mathrm{X}}$ formulae: $\psi_3 = \mathrm{EGF}(critical_1 \vee \ldots \vee critical_N)$, $\psi_4 = \mathrm{EF}(trying_1 \wedge \ldots \wedge trying_N \wedge \mathrm{G}(\neg critical_2 \wedge \ldots \wedge \neg critical_N))$.

The above systems have been carefully selected in order to reveal the advantages and disadvantages of both SAT- and BDD-based BMC approaches.









For the SAT-based BMC method two semantics are implemented: the alternating and the non-alternating one. The results obtained for the non-alternating semantics are superior to those for the alternating one in the following two cases: (1) the length of the witness and the number of k-paths depends on the number of components of the considered system; (2) the number of k-paths is constant and their lengths are at least twice as long in the alternating semantics as in the non-alternating one. On the other hand, the non-alternating semantics is inferior to the alternating one in the case when the length of k-paths is independent of the number of components of the considered system and their number is independent of their lengths. Further, the assumed time limit (1800s) prefers the non-alternating semantics, i.e., if a larger time limit than 1800s is set, then for the alternating semantics much more components of a given system can be verified than for the non-alternating one (see Mutex and the formula ψ_1). The reason is that the SAT-based BMC method for systems with a large number of components (for the non-alternating semantics) generates the propositional formulae that are more complicated than in case of the alternating semantics. This results in the fact that the memory consumed by the SAT-solver is significantly larger for the set of clauses generated in case of the non-alternating semantics, therefore only smaller systems can be model-checked.

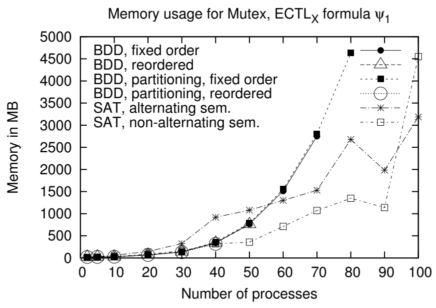
The method based on BDDs is implemented with reordering, and with the fixed interleaving order of the BDD variables. The reordering is performed by the Rudell's sifting algorithm available in the implementation of CUDD library. Moreover, we also use partitioned transition relations. In the case of GTPP, the BDD-based method is remarkably superior to the SAT-based method in terms of the verification times and the consumed memory for the tested formulae. The reason is the substantial number of k-paths in SAT-BMC, which causes a larger memory consumption and longer running times in comparison with the BDD-based method. Where the length of the witness is constant regardless of the number of the processes (i.e., in Mutex for ψ_1 and the corresponding formula ψ_3), the SAT-based method is more efficient than the BDD-based one. Our partitioning of the transition relation does not reduce noticeably the memory usage, although in most of the considered cases the method benefits from the reordering of the BDD variables. The BDD-based method deals better with the increasing length of the witness when scaling in the number of processes or nodes. In the case of Mutex, our experiments revealed that the method based on BDDs is more efficient for small and medium models, but it consumes more memory. The above observations seem to be consistent with other existing comparisons of SAT versus BDD [2].

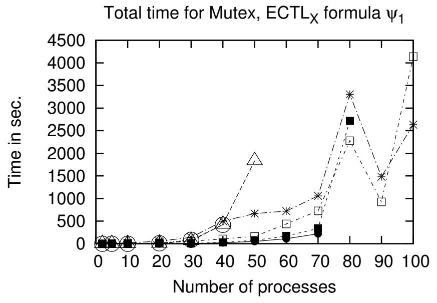
We compare also our results with those of Tina, however, as Tina does not support a verification of $\mathrm{ECTL}_{-\mathrm{X}}$ formulae, the results only for $\mathrm{ELTL}_{-\mathrm{X}}$ are taken into account. Unsurprisingly, as Tina is a non-bounded model checker, the results are inferior to the results of our BMC methods. Although Tina seems to perform well in the case of ψ_4 for Mutex, it suffers from a significant increase of the memory usage for 8 processes and is unable to verify more than 9 processes.

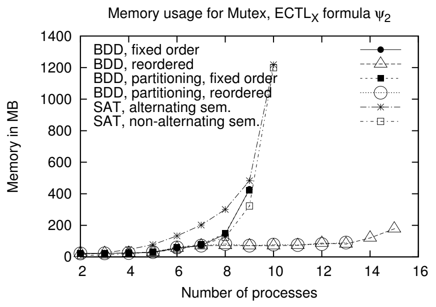
All the benchmarks can be found at the webpage of VerICS, together with an instruction how to reproduce our results. For the tests we have used a computer running Linux 2.6.38 with two Intel Xeon 2.00GHz processors and 4 GB of memory. Both the algorithms have been implemented in C++. The BDD-based method uses CUDD [30], which is a general purpose BDD library, while the SAT-based technique uses MiniSat2 [14] for testing satisfiability of the generated propositional formulae.

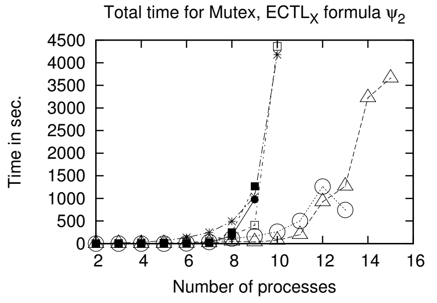
7 Conclusions

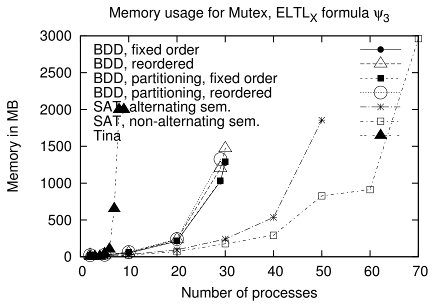
In this paper we have presented two different approaches for bounded model checking of DTPNs: via a reduction to SAT and via BDDs. The two methods have been tested and compared to each other on two standard benchmarks. The specifications were given in the $\mathrm{ECTL}_{-\mathrm{X}}$ and $\mathrm{ELTL}_{-\mathrm{X}}$ languages. Additionally we have compared our results with those obtained using the tool Tina. The experimental results revealed that SAT-based BMC and BDD-based BMC are complementary solutions to the BMC problem, as their performance depends

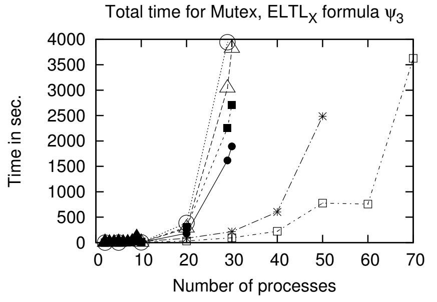

Memory usage for Mutex, ECTL$_X$ formula ψ_1
BDD, fixed order
BDD, reordered
BDD, partitioning, fixed order
BDD, partitioning, reordered
SAT, alternating sem.
SAT, non-alternating sem.
Memory in MB
Number of processes

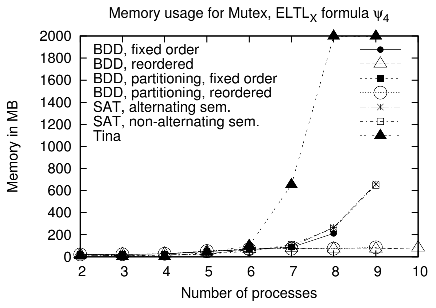

Total time for Mutex, ECTL$_X$ formula ψ_1
Time in sec.
Number of processes

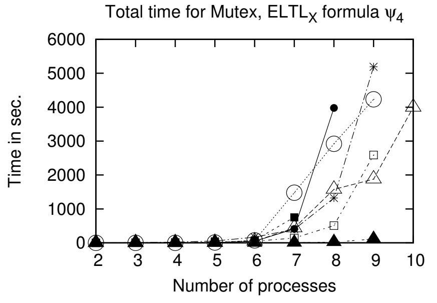

Memory usage for Mutex, ECTL$_X$ formula ψ_2
BDD, fixed order
BDD, reordered
BDD, partitioning, fixed order
BDD, partitioning, reordered
SAT, alternating sem.
SAT, non-alternating sem.
Memory in MB
Number of processes


Total time for Mutex, ECTL$_X$ formula ψ_2
Time in sec.
Number of processes


Memory usage for Mutex, ELTL$_X$ formula ψ_3
BDD, fixed order
BDD, reordered
BDD, partitioning, fixed order
BDD, partitioning, reordered
SAT, alternating sem.
SAT, non-alternating sem.
Tina
Memory in MB
Number of processes


Total time for Mutex, ELTL$_X$ formula ψ_3
Time in sec.
Number of processes


Memory usage for Mutex, ELTL$_X$ formula ψ_4
BDD, fixed order
BDD, reordered
BDD, partitioning, fixed order
BDD, partitioning, reordered
SAT, alternating sem.
SAT, non-alternating sem.
Tina
Memory in MB
Number of processes


Total time for Mutex, ELTL$_X$ formula ψ_4
Time in sec.
Number of processes

	SAT-BMC		**BDD-BMC**
	alternating sem.	**non-alternating sem.**	**non-alternating sem.**
Formula	$(\mathbf{k}, \mathbf{f_k}(\psi))$	$(\mathbf{k}, \mathbf{f_k}(\psi))$	**the number of iterations**
GPP, ψ_1	$(4 \cdot n + 6, 4 \cdot n + 8)$	$(2 \cdot n + 3, 2 \cdot n + 5)$	$2 \cdot n + 2$
GPP, ψ_2	$(4 \cdot n + 6, 1)$	$(2 \cdot n + 3, 1)$	$2 \cdot n + 2$
Mutex, ψ_1	$(8, 10)$	$(4, 6)$	4
Mutex, ψ_2	$(2 \cdot n + 8, 2)$	$(n + 2, 2)$	$2 \cdot n + 1$
Mutex, ψ_3	$(14, 1)$	$(6, 1)$	5
Mutex, ψ_4	$(4 \cdot n + 8, 1)$	$(2 \cdot n + 2, 1)$	$2 \cdot n + 1$

Table 1. The sizes of the witnesses. The number of nodes/processes is denoted by n.

on the system and the property that are verified. The approach based on BDDs scales better than the SAT-based one, when witnesses are found at small and constant depths with respect to the scaling parameter. From two of the considered semantics for SAT-BMC, the non-alternating one is more efficient.

The paper is the first one to present bounded model checking methods for verifying $\mathrm{ECTL}_{-\mathrm{X}}$ and $\mathrm{ELTL}_{-\mathrm{X}}$ properties of time Petri nets. The encodings that are used in the SAT-based method, are applied in the context of BMC and DTPNs for the first time. Similarly, the verification methods for $\mathrm{ECTL}_{-\mathrm{X}}$ and $\mathrm{ELTL}_{-\mathrm{X}}$ used in BDD-BMC have not been considered before in the bounded model checking of time Petri nets. The dependence on the length of the witnesses, and the performance of the two BMC methods for DTPNs has not been observed before as well.

As this is our early attempt at BDD-based bounded model checking, it suffers from some weaknesses. In particular, the encoding of the transition relation could be improved, and some more recent developments in BDD-based symbolic model checking could be applied.

In our future work we are going to consider dense semantics and more general time Petri nets.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. N. Amla, R. Kurshan, K. McMillan, and R. Medel. Experimental analysis of different techniques for bounded model checking. In *Proc. of TACAS'03*, volume 2619 of *LNCS*, pp. 34–48. Springer-Verlag, 2003.
3. G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. of FORTE'02*, volume 2529 of *LNCS*, pp. 243–259. Springer-Verlag, 2002.
4. M. Benedetti and A. Cimatti. Bounded model checking for Past LTL. In *Proc. of TACAS'03*, volume 2619 of *LNCS*, pp. 18–33. Springer-Verlag, 2003.
5. A. Biere, A. Cimatti, E. Clarke, M.Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. of DAC'99*, pp. 317–320, 1999.

6. H. Boucheneb, G. Gardey, and O. H. Roux. TCTL model checking of time Petri nets. *Journal of Logic and Computation*, 19(6):1509–1540, 2009.
7. H. Boucheneb and R. Hadjidj. CTL* model checking for time Petri nets. *Theoretical Computer Science*, 353(1):208–227, 2006.
8. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, 35(8):677–691, 1986.
9. J. R. Burch, E. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1990.
10. G. Cabodi, P. Camurati, and S. Quer. Can BDD compete with SAT solvers on bounded model checking? In *Proc. of DAC'02*, pp. 117–122, 2002.
11. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of CAV'94*, volume 818 of *LNCS*, pp. 415–427. Springer-Verlag, 1994.
12. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
13. F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Vardi. Benefits of bounded model checking at an industrial setting. In *Proc. of CAV'01*, volume 2102 of *LNCS*, pp. 436–453. Springer-Verlag, 2001.
14. N. Eén and N. Sörensson. MiniSat - A SAT Solver with Conflict-Clause Minimization. In *Proc. of SAT'05*, LNCS. Springer-Verlag, 2005.
15. G. Luettgen G. Ciardo and A. S. Miner. Exploiting interleaving semantics in symbolic state-space generation. *Formal Methods in System Design*, 31:63–100, 2007.
16. K. Heljanko and I. Niemelä. Bounded LTL model checking with stable models. In *Proc. of LPNMR'01*, volume 2173 of *LNCS*, pp. 200–212. Springer-Verlag, 2001.
17. M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.
18. R. Janicki. Nets, sequential components and concurrency relations. *Theoretical Computer Science*, 29:87–121, 1984.
19. A. Jones and A. Lomuscio. A BDD-based BMC approach for the verification of multi-agent systems. In *Proc. of CS&P'09*, volume 1, pp. 253–264. Warsaw University, 2009.
20. D. Lime and O. H. Roux. Model checking of time Petri nets using the state class timed automaton. *Discrete Event Dynamic Systems*, 16(2):179–205, 2006.
21. K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
22. P. Merlin and D. J. Farber. Recoverability of communication protocols – implication of a theoretical study. *IEEE Trans. on Communications*, 24(9):1036–1043, 1976.
23. A. Męski, W. Penczek, and A. Półrola. BDD-based bounded model checking for elementary net systems. In *Proc. of CS&P'10*, volume 237(1) of *Informatik-Berichte*, pp. 219–230. Humboldt University, 2010.
24. A. Miner and G. Ciardo. Efficient reachability set generation and storage using decision diagrams. In *Proc. of ICATPN'99*, volume 1639 of *LNCS*, pp. 6–25. Springer-Verlag, 1999.
25. D. Peled. All from one, one for all: On model checking using representatives. In *Proc. of CAV'93*, volume 697 of *LNCS*, pp. 409–423. Springer-Verlag, 1993.
26. W. Penczek and A. Półrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*. Springer-Verlag, 2006.
27. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.

28. Knot Pipatsrisawat and Adnan Darwiche. Rsat 2.0: Sat solver description. Technical Report D–153, Automated Reasoning Group, Computer Science Department, UCLA, 2007.
29. A. Półrola and W. Penczek. Minimization algorithms for time Petri nets. *Fundamenta Informaticae*, 60(1-4):307–331, 2004.
30. F. Somenzi. CUDD: CU decision diagram package - release 2.3.1. http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html.
31. M. Wan and G. Ciardo. Symbolic reachability analysis of integer timed petri nets. In *Proc. of SOFSEM'2009*, pp. 595–608, 2009.
32. B. Woźna. ACTL* properties and bounded model checking. *Fundamenta Informaticae*, 63(1):65–87, 2004.
33. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for timed automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
34. A. J. Yu, G. Ciardo, and G. Luettgen. Decision-diagram-based techniques for bounded reachability checking of asynchronous systems. *Software Tools for Technology Transfer*, 11(2):117–131, 2009.
35. A. Zbrzezny. Improving the translation from ECTL to SAT. *Fundamenta Informaticae*, 85(1-4):513–531, 2008.

A Appendix: Models for DTPNs - Proofs

In order to show that $\widehat{M}_c(\mathcal{N})$ can replace $M_c(\mathcal{N})$ in $\mathrm{CTL}^*_{-\mathrm{X}}$verification (i.e., to prove Lemma 1) we shall prove the following lemma:

Lemma 3. *For a given distributed time Petri net $\mathcal{N}$ the models $M_c(\mathcal{N}) = (T \cup \mathbb{N}, \Sigma, \sigma^0, \rightarrow_c), V_c$ and $\widetilde{M}_c(\mathcal{N}) = (T \cup \mathbb{C}_{\mathcal{N}}, \Sigma, \sigma^0, \rightarrow_r, V_c)$ are bisimulation equivalent.*

Proof. We shall show that the relation $\mathcal{R} = \{((m, clock), (m', clock')) \mid m = m' \wedge \forall(\mathfrak{i} \in \mathfrak{I} \text{ s.t. } clock(\mathfrak{i}) \leq c_{max}(\mathcal{N}))\ clock(\mathfrak{i}) = clock'(\mathfrak{i}) \wedge \forall(\mathfrak{i} \in \mathfrak{I} \text{ s.t. } clock(\mathfrak{i}) > c_{max}(\mathcal{N}))\ clock'(\mathfrak{i}) > c_{max}(\mathcal{N})\}$ is a bisimulation. It is easy to see that $\sigma^0 \mathcal{R} \sigma^0$, and the valuations of the related states are equal (due to equality of their markings). Consider $\sigma = (m, clock) \in \Sigma$ and $\sigma' = (m, clock') \in \Sigma$ such that $\sigma \mathcal{R} \sigma'$.

- if $\sigma \overset{\delta}{\rightarrow}_c \sigma_1$, where $\delta \in \mathbb{N}$, then for each $t \in en(m)$ there exists $\mathfrak{i} \in \mathcal{IV}(t)$ s.t. $clock(\mathfrak{i}) + \delta \leq Lft(t)$. Consider the following cases:
  - if $en(m)$ contains at least one transition t with $Lft(t) < \infty$, then this implies that $\delta \leq c_{max}(\mathcal{N})$. In this case consider $\delta' = \delta$; it is easy to see from the definition of $\mathcal{R}$ that for any $t \in en(m)$ s.t $Lft(t) < \infty$ if in σ for some $\mathfrak{i} \in \mathfrak{I}$ we have $clock(\mathfrak{i}) + \delta \leq Lft(t)$, then in σ' $clock'(\mathfrak{i}) + \delta' \leq Lft(t)$ holds as well, and therefore the time δ' can pass at σ', leading to the state $\sigma' + \delta'$, which satisfies $(\sigma + \delta)\mathcal{R}(\sigma' + \delta')$ in an obvious way.
  - if $en(m)$ contains no transition t with $Lft(t) < \infty$, then we can have either $\delta < c_{\mathbf{m1}}$ or $\delta \geq c_{\mathbf{m1}}$, where by $c_{\mathbf{m1}}$ we mean the value $c_{max}(\mathcal{N})+1$. In the first case consider $\delta' = \delta$; it is obvious that such a passage of time at σ' disables no transition and is allowed therefore; it is also easy to see

that $(\sigma + \delta)\mathcal{R}(\sigma' + \delta')$. In the case $\delta \geq c_{\mathbf{m1}}$ assume $\delta' = c_{\mathbf{m1}}$. Again, it is obvious that such a passage of time at σ' disables no transition and due to this is allowed, and that in both the states $\sigma + \delta$ and $\sigma' + \delta'$ we have $clock(\mathfrak{i}) > c_{max}(\mathcal{N})$ for all $i \in \mathfrak{I}$, and therefore $(\sigma + \delta)\mathcal{R}(\sigma' + \delta')$.

– the three remaining cases are straightforward.

Next, we can prove Lemma 1:

Proof. To prove the lemma, it is sufficient to show that $\widetilde{M}_c(\mathcal{N})$ and $\widehat{M}_c(\mathcal{N})$ are bisimulation equivalent. So, we shall show that the relation $\mathcal{R} \subseteq \Sigma \times \Sigma_s$ given by $\mathcal{R} = \{((m, clock), (m', clock_s)) \mid m = m' \wedge clock_s = clock|_{c_{\mathbf{m1}}}\}$ is a bisimulation. It is easy to see that $\sigma^0 \mathcal{R} \sigma^0$, and that the valuations of the related states are equal (due to equality of their markings). Consider $\sigma = (m, clock) \in \Sigma$ and $\sigma' = (m, clock_s) \in \Sigma_s$ with $clock_s = clock|_{c_{\mathbf{m1}}}$.

– if $\sigma \xrightarrow{\delta}_r \sigma_1$, where $\delta \in \mathbb{C}_{\mathcal{N}}$, then for each $t \in en(m)$ there exists $\mathfrak{i} \in I\mathcal{V}(t)$ s.t. $clock(\mathfrak{i}) + \delta \leq Lft(t)$. Due to the fact that for each $\mathfrak{i} \in \mathfrak{I}$ it holds $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \leq clock(\mathfrak{i})$, the time δ can pass at σ' as well, leading to the state $(m, clock|_{c_{\mathbf{m1}}} \oplus \delta)$. Consider the states $(m, clock + \delta)$ and $(m, clock|_{c_{\mathbf{m1}}} \oplus \delta)$; we should show that $(clock + \delta)|_{c_{\mathbf{m1}}} = clock|_{c_{\mathbf{m1}}} \oplus \delta$. We have the following cases: if $clock(\mathfrak{i}) = clock|_{c_{\mathbf{m1}}}(\mathfrak{i})$ and $clock(\mathfrak{i}) + \delta < c_{\mathbf{m1}}$, then $clock(\mathfrak{i}) + \delta = clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) + \delta = clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \oplus \delta$. If $clock(\mathfrak{i}) = clock|_{c_{\mathbf{m1}}}(\mathfrak{i})$ and $clock(\mathfrak{i}) + \delta \geq c_{\mathbf{m1}}$ then $clock|_{c_{\mathbf{m1}}}(\mathfrak{i})$
$\oplus\, \delta = c_{\mathbf{m1}}$, and therefore $(clock + \delta)|_{c_{\mathbf{m1}}}(\mathfrak{i}) = clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \oplus \delta$. If $clock(\mathfrak{i}) \geq c_{\mathbf{m1}}$ and $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) = c_{\mathbf{m1}}$ then $clock(\mathfrak{i}) + \delta \geq c_{\mathbf{m1}}$ and $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \oplus \delta = c_{\mathbf{m1}} = (clock + \delta)|_{c_{\mathbf{m1}}}(\mathfrak{i})$, which ends this part of the proof.

– if $\sigma' \xrightarrow{\delta}_s \sigma_1'$, where $\delta \in \mathbb{C}_{\mathcal{N}}$ then for each $t \in en(m)$ there exists $\mathfrak{i} \in I\mathcal{V}(t)$ s.t. $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \oplus \delta \leq Lft(t)$. If $Lft(t) < \infty$, then this implies $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \oplus \delta \leq c_{max}(\mathcal{N})$, which in turn gives that $clock|_{c_{\mathbf{m1}}}(\mathfrak{i}) \leq c_{max}(\mathcal{N})$, and therefore $clock(\mathfrak{i}) = clock|_{c_{\mathbf{m1}}}(\mathfrak{i})$, $clock(\mathfrak{i}) + \delta \leq c_{max}(\mathcal{N})$ and finally $clock(\mathfrak{i}) + \delta \leq Lft(t)$, while if $Lft(t) = \infty$ then $clock(\mathfrak{i}) + \delta \leq Lft(t)$ in an obvious way. Thus, the time δ can pass in σ as well. Consider the states $(m, clock + \delta)$ and $(m, clock|_{c_{\mathbf{m1}}} \oplus \delta)$; we should show that $(clock + \delta)|_{c_{\mathbf{m1}}} = clock|_{c_{\mathbf{m1}}} \oplus \delta$, which can be done analogously as in the previous part of the proof.

– The remaining two cases are straightforward.

Finally, we prove that the relation $\simeq_\star$ preserves the behaviours of the net (Lemma 2):

Proof. Consider the states $\sigma = (m, clock)$ and $\sigma' = (m, clock')$ $(\sigma, \sigma' \in \Sigma)$ s.t. $\sigma \simeq_\star \sigma'$.

– Consider $l = \delta \in \mathbb{N}$. The time δ can pass in σ iff for each $t \in en(m)$ there is $\mathfrak{i} \in I\mathcal{V}(t)$ s.t. $clock(\mathfrak{i}) + \delta \leq Lft(t)$. If $Lft(t) < \infty$, then we have that $\min_{\mathfrak{i} \in I\mathcal{V}(t)} clock(\mathfrak{i}) + \delta \leq Lft(t) \leq c_{max}(\mathcal{N})$, which implies that the states σ, σ' satisfy $\min_{\mathfrak{i} \in I\mathcal{V}(t)} clock(i) = \min_{\mathfrak{i} \in I\mathcal{V}(t)} clock'(i)$, and in turn $\min_{\mathfrak{i} \in I\mathcal{V}(t)} clock'(\mathfrak{i}) + \delta = \min_{\mathfrak{i} \in I\mathcal{V}(t)} clock(\mathfrak{i}) + \delta \leq Lft(t)$. If $Lft(t) = \infty$ we can have two cases: if $\min_{\mathfrak{i} \in I\mathcal{V}(t)} clock(i) = \min_{\mathfrak{i} \in I\mathcal{V}(t)} clock'(i) \leq c_{max}(\mathcal{N})$ then $\min_{\mathfrak{i} \in I\mathcal{V}(t)} clock(i) + \delta = \min_{\mathfrak{i} \in I\mathcal{V}(t)} clock'(i) + \delta$ which is not greater than $Lft(t)$ in an obvious way, while if $\min_{\mathtt{i} \in IV(t)} clock(i) > c_{max}(\mathcal{N})$ and $\min_{\mathtt{i} \in IV(t)} clock'(i) > c_{max}(\mathcal{N})$ then both $\min_{\mathtt{i} \in IV(t)} clock(i) + \delta$ and $\min_{\mathtt{i} \in IV(t)} clock'(i) + \delta$ are greater than $c_{max}(\mathcal{N})$ and do not exceed $Lft(t)$. Thus, the same time can pass at σ and at σ', and the obtained states are $\star$-equivalent.

- Consider $l = t \in T$ such that $t \in en(m)$. The transition t can fire at σ leading to a state $\sigma_1 = (m_1, clock_1)$ iff for each $\mathtt{i} \in IV(t)$ we have $clock(\mathtt{i}) \geq Eft(t)$ and there is $\mathtt{i} \in IV(t)$ such that $clock(\mathtt{i}) \leq Lft(t)$.
  - If $Lft(t) < \infty$ then from $\sigma \simeq_\star \sigma'$ we have that $\min_{\mathtt{i} \in IV(t)} clock(i) = \min_{\mathtt{i} \in IV(t)} clock'(i)$, which implies that for each $\mathtt{i} \in IV(t)$ $clock'(\mathtt{i}) \geq Eft(t)$ and there is $\mathtt{i} \in IV(t)$ such that $clock'(\mathtt{i}) \leq Lft(t)$, which means that t can fire at σ' as well, leading to a state $\sigma_1' = (m_1', clock_1')$. In the obtained states we have $m_1 = m_1'$, $clock_1(\mathtt{i}) = 0 = clock_1'(\mathtt{i})$ for each $\mathtt{i} \in IV(t)$, and $clock_1(\mathtt{i}) = clock(\mathtt{i})$, $clock_1'(\mathtt{i}) = clock'(\mathtt{i})$ otherwise. Consider a transition $t' \in en(m')$. If $IV(t) \cap IV(t') \neq \emptyset$ then $\min_{\mathtt{i} \in IV(t')} clock_1(i) = \min_{\mathtt{i} \in IV(t')} clock_1'(i) = 0$, while if $IV(t) \cap IV(t') = \emptyset$ then for each $i \in IV(t')$ the relation between $clock_1(\mathtt{i})$ and $clock_1'(\mathtt{i})$ is the same as between $clock(\mathtt{i})$ and $clock'(\mathtt{i})$, which implies that either $\min_{\mathtt{i} \in IV(t')} clock_1(i) = \min_{\mathtt{i} \in IV(t')} clock_1'(i) \leq c_{max}(\mathcal{N})$ or $\min_{\mathtt{i} \in IV(t')} clock_1(i) > c_{max}(\mathcal{N}) \wedge \min_{\mathtt{i} \in IV(t')} clock_1'(i) > c_{max}(\mathcal{N})$. Thus, we have $\sigma_1 \simeq_\star \sigma_1'$.
  - If $Lft(t) = \infty$ then from the definition of $c_{max}(\mathcal{N})$ we have that $Eft(t) \geq c_{max}(\mathcal{N})$, and therefore from the definition of $\simeq_\star$ for each $\mathtt{i} \in IV(t)$ it holds $clock'(\mathtt{i}) \geq Eft(t)$, while for all $\mathtt{i} \in IV(t)$ $clock'(t) < Lft(t)$ in an obvious way. Thus, the transition can fire at σ' as well, leading to a state $\sigma_1' = (m_1, clock_1')$. The proof that $\sigma_1 \simeq_\star \sigma_1'$ is analogous to the case $Lft(t) < \infty$.
- The rest of the proof is straightforward.