Bounded Parametric Model Checking for Petri Nets

Wojciech Penczek

a joint work with Michał Knapik and Agata Półrola

Institute of Computer Science, Polish Academy of Sciences

Carl Adam Petri Memorial Symposium, Berlin, 4 February 2011
Outline

1. Introduction to Parametric Model Checking
2. Benchmark: Mutual exclusion (MUTEX)
3. Syntax and Semantics of PRTCTL
4. Bounded parametric model checking for ENS
5. Parametric reachability for DTPN
6. Experimental Results
7. Final Remarks
A Kripke model $M$ satisfies a modal formula $\phi$.

For Petri Nets

$M$ is a model corresponding either to the marking graph of an EPN or to the concrete state graph of a TPN.
Parametric Model Checking

Parameters can appear in:

- a (timed) model\(^1,5\)
- a formula\(^2,3\)
- a model and a formula\(^4\)

\[\forall \Theta \leq b \; EF(\neg p \land EG \leq \Theta c_1)\]


Parametric Model Checking

Parameters can appear in:

- a (timed) model\(^1,5\)
- a formula\(^2,3\)
- a model and a formula\(^4\)

\[
\forall \Theta \leq b \ EF(\neg p \land EG^{\leq \Theta} c_1)
\]

---

Parametric Model Checking

Parameters can appear in:

- a (timed) model\(^1,^5\)
- a formula\(^2,^3\)
- a model and a formula\(^4\)

\[
\forall \Theta \leq b \ EF(\neg p \land \ EG \leq \Theta \ c_1)
\]

Time Petri Net: Timed Mutex

![Time Petri Net Diagram](image)

The diagram represents a Time Petri Net with timed mutex mechanisms. It includes places and transitions with timing labels. The states and transitions are as follows:

- **Places:**
  - Place 0
  - Place 1
  - Place 2
- **Transitions:**
  - Start1
  - Start2
  - Trying1
  - Trying2
  - Setx1
  - Setx2
  - Enter1
  - Enter2
  - Waiting1
  - Waiting2
  - Setx1_copy1
  - Setx2_copy2

The timing labels are given in the form of intervals, indicating the time delays or durations.
Parametric Model Checking

Complexity
If parameters are in:
- a model (e.g., TA, TPN), then reachability is undecidable,
- a formula, then for TECTL – $3\text{EXPTIME}$,
- both a model and a formula, then reachability is undecidable.

Idea
SAT-based Bounded Model Checking applied to parametric verification.

Applications
BMC for PRTCTL\(^1\):
- parameters in formulas for Elementary Petri Nets\(^2\), and
- parametric reachability for Time Petri Nets\(^3\).

Parametric Model Checking

Complexity

If parameters are in:

- a model (e.g., TA, TPN), then reachability is undecidable,
- a formula, then for TECTL – $3\text{EXPTIME}$,
- both a model and a formula, then reachability is undecidable.

Idea

SAT-based Bounded Model Checking applied to parametric verification.

Applications

BMC for PRTCTL$^1$:

- parameters in formulas for Elementary Petri Nets$^2$, and
- parametric reachability for Time Petri Nets$^3$.

---

Parametric Model Checking

Complexity
If parameters are in:
- a model (e.g., TA, TPN), then reachability is undecidable,
- a formula, then for TECTL – $3\text{EXPTIME}$,
- both a model and a formula, then reachability is undecidable.

Idea
SAT-based Bounded Model Checking applied to parametric verification.

Applications
BMC for PRTCTL$^1$:
- parameters in formulas for Elementary Petri Nets$^2$, and
- parametric reachability for Time Petri Nets$^3$.

---


Working example: mutual exclusion

Petri Net: MUTEX

Mutual exclusion:
- \( n \) processes compete for access to the shared resource \( p \),
- **token in:**
  - \( w_i \): the \( i \)-th process is waiting,
  - \( c_i \): the \( i \)-th process in a critical section,
  - \( r_i \): the \( i \)-th process is in an unguarded section,
  - \( p \): the resource is available.
Syntax and Semantics

Syntax of $v$RTCTL

- $\mathcal{PV}$ – propositional formulas, containing the symbol $true$,
- $Parameters = \{\Theta_1, \ldots, \Theta_n\}$ – parameter variables,
- $Linear expressions - \eta = \sum_{i=1}^{n} c_i \Theta_i + c_0$, where $c_0, \ldots, c_n \in \mathbb{N}$.

$v$RTCTL syntax:

- $\mathcal{PV} \subseteq v$RTCTL,
- if $\alpha, \beta \in v$RTCTL, then $\neg \alpha, \alpha \lor \beta, \alpha \land \beta \in v$RTCTL,
- if $\alpha, \beta \in v$RTCTL, then $EX \alpha, EG \alpha, E \alpha U \beta \in v$RTCTL,
- if $\alpha, \beta \in v$RTCTL, then $EG^{\leq \eta} \alpha, E \alpha U^{\leq \eta} \beta \in v$RTCTL.

Example

$\varphi(\Theta) = EF(\neg p \land EG^{\leq \Theta} c_1)$

$(EF\alpha = EtrueU\alpha$ – a derived modality)
Syntax and Semantics

Syntax of $\nu$RTCTL

- $\mathcal{PV}$ – propositional formulas, containing the symbol $true$,
- $Parameters = \{\Theta_1, \ldots, \Theta_n\}$ – parameter variables,
- Linear expressions $- \eta = \sum_{i=1}^{n} c_i \Theta_i + c_0$, where $c_0, \ldots, c_n \in \mathbb{N}$.

$\nu$RTCTL syntax:

- $\mathcal{PV} \subseteq \nu$RTCTL,
- if $\alpha, \beta \in \nu$RTCTL, then $\neg \alpha, \alpha \lor \beta, \alpha \land \beta \in \nu$RTCTL,
- if $\alpha, \beta \in \nu$RTCTL, then $EX \alpha, EG \alpha, E \alpha U \beta \in \nu$RTCTL,
- if $\alpha, \beta \in \nu$RTCTL, then $EG^{\leq \eta} \alpha, E \alpha U^{\leq \eta} \beta \in \nu$RTCTL.

Example

$$\varphi(\Theta) = EF(\neg p \land EG^{\leq \Theta} c_1)$$

($EF \alpha = EtrueU \alpha$ – a derived modality)
Syntax and semantics

Model for $\nu$RTCTL and PRTCTL

A Kripke structure $M = (S, \rightarrow, \mathcal{L})$ is a model, where

- $S$ – a finite set of states,
- $\rightarrow \subseteq S \times S$ – a transition relation s.t. $\forall s \in S \exists s' \in S \ s \rightarrow s'$,
- $\mathcal{L} : S \rightarrow 2^{\mathbb{P}V}$ – a labelling function s.t. $\forall s \in S \ true \in \mathcal{L}(s)$.

Parameter valuations

$\nu$RTCTL formulae are interpreted under parameter valuations:

- $\nu : Parameters \rightarrow \mathbb{N}$,
- $\nu$ is extended to the linear expressions $\eta$.

Example

For $\varphi(\Theta) = EF(\neg p \land EG^{\leq \Theta} c_1)$ and $\nu$ s.t. $\nu(\Theta) = 2$

$\varphi(\nu) = EF(\neg p \land EG^{\leq 2} c_1)$
Syntax and semantics

Model for \( \nuRTCTL \) and \( \nuPRTCTL \)

A Kripke structure \( M = (S, \rightarrow, \mathcal{L}) \) is a model, where

- \( S \) – a finite set of states,
- \( \rightarrow \subseteq S \times S \) – a transition relation s.t. \( \forall s \in S \exists s' \in S \ s \rightarrow s' \),
- \( \mathcal{L} : S \rightarrow 2^{\nu} \) – a labelling function s.t. \( \forall s \in S \ true \in \mathcal{L}(s) \).

Parameter valuations

\( \nuRTCTL \) formulae are interpreted under parameter valuations:

- \( \nu : Parameters \rightarrow \mathbb{N} \),
- \( \nu \) is extended to the linear expressions \( \eta \).

Example

For \( \varphi(\Theta) = EF(\neg p \land EG^{\leq \Theta} c_1) \) and \( \nu \) s.t. \( \nu(\Theta) = 2 \)

\[ \varphi(\nu) = EF(\neg p \land EG^{\leq 2} c_1) \]
Syntax and semantics

**Model for \(vRTCTL\) and \(PRTCTL\)**

A Kripke structure \(M = (S, \rightarrow, \mathcal{L})\) is a model, where

- \(S\) – a finite set of states,
- \(\rightarrow \subseteq S \times S\) – a transition relation s.t. \(\forall s \in S \exists s' \in S\) \(s \rightarrow s'\),
- \(\mathcal{L} : S \rightarrow 2^\mathcal{P}\nu\) – a labelling function s.t. \(\forall s \in S\) true \(\in \mathcal{L}(s)\).

**Parameter valuations**

\(vRTCTL\) formulae are interpreted under parameter valuations:

- \(\nu : Parameters \rightarrow \mathbb{N}\),
- \(\nu\) is extended to the linear expressions \(\eta\).

**Example**

For \(\varphi(\Theta) = EF(\neg p \land EG^{\leq \Theta} c_1)\) and \(\nu\) s.t. \(\nu(\Theta) = 2\)
\[
\varphi(\nu) = EF(\neg p \land EG^{\leq 2} c_1)
\]
Syntax and semantics

\[ M, \text{start} \models E G^{\leq 3} \alpha \]

\[ M, \text{start} \models E \alpha U^{\leq 3} \beta \]
Syntax and Semantics

\[ M, \text{start} \models \text{EX}_\alpha \]

\[ M, \text{start} \models \text{EF}^{\leq 2}_\alpha \]
Syntax

**Syntax of PRTCTL**

- $\nu RTCTL \subseteq PRTCTL$,
- If $\alpha(\Theta) \in \nu RTCTL \cup PRTCTL$, then
  $\forall_\Theta \alpha(\Theta), \exists_\Theta \alpha(\Theta), \forall_\Theta \leq a \alpha(\Theta), \exists_\Theta \leq a \alpha(\Theta) \in PRTCTL$ for $a \in \mathbb{N}$.

Notation: $\alpha(\Theta_1, \ldots, \Theta_n)$ denotes that $\Theta_1, \ldots, \Theta_n$ are free in $\alpha$.

**Example**

$\varphi_1^3 = \forall_\Theta \leq 3 EF(\neg p \land EG \leq \Theta c_1)$

We consider the closed formulae (sentences) of PRTCTL only.
Syntax of PRTCTL

- \( \nu RTCTL \subseteq PRTCTL \),
- if \( \alpha(\Theta) \in \nu RTCTL \cup PRTCTL \), then
  \( \forall \Theta \alpha(\Theta), \exists \Theta \alpha(\Theta), \forall \Theta \leq a \alpha(\Theta), \exists \Theta \leq a \alpha(\Theta) \in PRTCTL \) for \( a \in \mathbb{N} \).

Notation: \( \alpha(\Theta_1, \ldots, \Theta_n) \) denotes that \( \Theta_1, \ldots, \Theta_n \) are free in \( \alpha \).

Example

\[ \varphi^3_1 = \forall \Theta \leq 3 EF(\neg p \land EG^{\leq \Theta} c_1) \]

We consider the closed formulae (sentences) of PRTCTL only.
Semantics

Semantics of \textit{PRTCTL} (the closed formulae)

- $M, s \models \forall \Theta \alpha(\Theta)$ iff $\bigwedge_{0 \leq i_\Theta} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \forall \Theta \leq a \alpha(\Theta)$ iff $\bigwedge_{0 \leq i_\Theta \leq a} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \exists \Theta \alpha(\Theta)$ iff $\bigvee_{0 \leq i_\Theta} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \exists \Theta \leq a \alpha(\Theta)$ iff $\bigvee_{0 \leq i_\Theta \leq a} M, s \models \alpha(\Theta \leftarrow i_\Theta)$.

Example

$M, s \models \varphi^3_1$ iff $\bigwedge_{i_\Theta \leq 3} M, s \models EF(\neg p \land EG \leq i_\Theta c_1)$
Semantics

Semantics of PRTCTL (the closed formulae)

- $M, s \models \forall \Theta \alpha(\Theta)$ iff $\land_{0 \leq i_{\Theta}} M, s \models \alpha(\Theta \leftarrow i_{\Theta})$,
- $M, s \models \forall \Theta \leq a \alpha(\Theta)$ iff $\land_{0 \leq i_{\Theta} \leq a} M, s \models \alpha(\Theta \leftarrow i_{\Theta})$,
- $M, s \models \exists \Theta \alpha(\Theta)$ iff $\lor_{0 \leq i_{\Theta}} M, s \models \alpha(\Theta \leftarrow i_{\Theta})$,
- $M, s \models \exists \Theta \leq a \alpha(\Theta)$ iff $\lor_{0 \leq i_{\Theta} \leq a} M, s \models \alpha(\Theta \leftarrow i_{\Theta})$.

Example

$M, s \models \varphi_1^3$ iff $\land_{i_{\Theta} \leq 3} M, s \models EF(\neg p \land EG^{\leq i_{\Theta}} c_1)$
Example of a PRTCTL formula

$$\forall \Theta [\text{AG}(\text{request} \Rightarrow AF \leq \Theta \text{receive}) \Rightarrow \text{AG}(\text{request} \Rightarrow AF \leq 2 \times \Theta \text{grant})]$$

expresses much more than the corresponding CTL formula

$$[\text{AG}(\text{request} \Rightarrow AF\text{receive}) \Rightarrow \text{AG}(\text{request} \Rightarrow AF\text{grant})]$$
Complexity of model checking

For **CTL, vRTCTL, and PRTCTL**

- **CTL** and **vRTCTL** can be model checked in time $O(|M| \cdot |\varphi|)$.

- **PRTCTL** can be model checked in time $O(|M|^{k+1} \cdot |\varphi|)$, where $k$ is the number of parameters in $\varphi$.

Existential fragments

The logics $v$RTECTL and PRTECTL are defined as the restrictions of, respectively, $v$RTCTL and the set of sentences of PRTCTL such that the negation can be applied to propositions only.

Example: $\varphi_1^4 = \forall_{\Theta \leq 4} EF(\neg p \land EG_{\leq \Theta} c_1)$
Existential fragments

The logics $v^{\text{RTECTL}}$ and $P^{\text{RTECTL}}$ are defined as the restrictions of, respectively, $v^{\text{RTCTL}}$ and the set of sentences of $P^{\text{RCTL}}$ such that the negation can be applied to propositions only.

Example: $\varphi_1^4 = \forall \Theta \leq 4 EF(\neg p \land EG \leq \Theta c_1)$
Let $\varphi_1^b = \forall_{\Theta \leq b} EF(\neg p \land EG_{\leq \Theta} c_1)$.

Intuitive meaning of $M, start \models \varphi_1^b$:

"There exists a future state, such that the resource is taken and the first process stays in the critical section for any time value bounded by $b"
**Bounded semantics**

### $k$–models

Idea – to unwind the computation tree of a model $M$ up to depth $k$.

- $M$ – a model, $k \in \mathbb{N}$,
- $Path_k$ – the set of all sequences $(s_0, \ldots, s_k)$, where $s_i \rightarrow s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the $k$-model.
- If an existential formula $\varphi$ holds in $M_k$, then $\varphi$ holds in $M$.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \land [\varphi]$ using a SAT-solver.
Bounded semantics

\(k\)-models

Idea – to unwind the computation tree of a model \(M\) up to depth \(k\).

- \(M\) – a model, \(k \in \mathbb{N}\),
- \(Path_k\) – the set of all sequences \((s_0, \ldots, s_k)\),
  where \(s_i \rightarrow s_{i+1}\).
- \(M_k = (Path_k, \mathcal{L})\) is called the \(k\)-model.
- If an existential formula \(\varphi\) holds in \(M_k\), then \(\varphi\) holds in \(M\).
- The problem \(M_k \models \varphi\) is translated to checking satisfiability of the propositional formula \([M_k] \land [\varphi]\) using a SAT-solver.
**Bounded semantics**

### $k$–models

Idea – to unwind the computation tree of a model $M$ up to depth $k$.

- $M$ – a **model**, $k \in \mathbb{N}$,
- $Path_k$ – the set of all sequences $(s_0, \ldots, s_k)$, where $s_i \rightarrow s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the **$k$-model**.
- If an existential formula $\varphi$ holds in $M_k$, then $\varphi$ holds in $M$.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \land [\varphi]$ using a SAT-solver.
Bounded semantics

$k$–models

Idea – to unwind the computation tree of a model $M$ up to depth $k$.

- $M$ – a model, $k \in \mathbb{N}$,
- $Path_k$ – the set of all sequences $(s_0, \ldots, s_k)$, where $s_i \rightarrow s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the \textit{k-model}.
- If an existential formula $\varphi$ holds in $M_k$, then $\varphi$ holds in $M$.

The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \land [\varphi]$ using a SAT-solver.
**Bounded semantics**

### $k$–models

Idea – to unwind the computation tree of a model $M$ up to depth $k$.

- $M$ – a model, $k \in \mathbb{N}$,
- $Path_k$ – the set of all sequences $(s_0, \ldots, s_k)$, where $s_i \rightarrow s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the $k$-model.
- If an existential formula $\varphi$ holds in $M_k$, then $\varphi$ holds in $M$.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \land [\varphi]$ using a SAT-solver.
Translation to boolean formula

Encoding submodels

\[ [M]_k^A := \bigwedge_{j \in A} \bigwedge_{i=0}^{k-1} T(w_i, j, w_{i+1}, j) \]

Where \( A \) – a set of path indices determined by function\(^5 f_k \).

\[ V \models [M]_k^A \text{ iff } V \text{ encodes } k\text{-model} \]

Encoding formulae

\( \varphi \) – a PRTCTL formula

\( \downarrow \)

\( [\varphi]_k \) – a propositional formula

Testing formula

\[ [M]_k^{F_k(\alpha)} \land I_s(w_{0,0}) \land [\varphi]_k \]

Translation to boolean formula

Encoding submodels

\[ [M]^A_k := \bigwedge_{j \in A} \bigwedge_{i=0}^{k-1} T(w_{i,j}, w_{i+1,j}) \]

Where \( A \) – a set of path indices determined by function\(^5\) \( f_k \).

\[ V \models [M]^A_k \text{ iff } V \text{ encodes } k\text{-model} \]

Encoding formulae

\( \varphi \) – a PRTCTL formula

\[ \downarrow \]

\[ [\varphi]^k \] – a propositional formula

Testing formula

\[ [M]^{F_k(\alpha)}_k \land I_s(w_{0,0}) \land [\varphi]_k \]

## Translation to boolean formula

### Encoding submodels

\[
[M]_k^A := \bigwedge_{j \in A} \bigwedge_{i=0}^{k-1} T(w_{i,j}, w_{i+1,j})
\]

Where \( A \) – a set of path indices determined by function\(^5\) \( f_k \).

\[ V \models [M]_k^A \text{ iff } V \text{ encodes } k\text{-model} \]

### Encoding formulae

\( \varphi \) – a PRTCTL formula

\[
[\varphi]_k \text{ – a propositional formula}
\]

### Testing formula

\[
[M]_{F_k(\alpha)}^k \land I_s(w_{0,0}) \land [\varphi]_k
\]

---

Distributed Time Petri Nets

Time Petri Nets

A Time Petri Net (TPN) - a tuple $N = (P, T, F, m_0, Eft, Lft)$, where:

- $P, T, F, m_0$ - like before,

- $Eft : T \rightarrow \mathbb{N}$, $Lft : T \rightarrow \mathbb{N} \cup \{\infty\}$ - earliest and latest firing times of transitions ($Eft(t) \leq Lft(t)$ for each $t \in T$)

Distributed Time Petri Nets

A Distributed Time Petri Net (DTPN) - a set of sequential\(^\ast\) TPNs, of pairwise disjoint sets of places, and communicating via joint transitions.

\(^\ast\) a net is sequential if none of its reachable markings concurrently enables two transitions
### Time Petri Nets

A Time Petri Net (TPN) - a tuple \( N = (P, T, F, m_0, Eft, Lft) \), where:

- \( P, T, F, m_0 \) - like before,
- \( Eft : T \rightarrow \mathbb{N} \), \( Lft : T \rightarrow \mathbb{N} \cup \{\infty\} \) - earliest and latest firing times of transitions (\( Eft(t) \leq Lft(t) \) for each \( t \in T \))

### Distributed Time Petri Nets

A Distributed Time Petri Net (DTPN) - a set of sequential (*) TPNs, of pairwise disjoint sets of places, and communicating via joint transitions.

(*) a net is sequential if none of its reachable markings concurrently enables two transitions
Example: Fischer’s mutual exclusion protocol

![Diagram of Fischer's mutual exclusion protocol](image)

W. Penczek (ICS PAS)  Parametric BMC for Petri Nets  the 4th of February, 2011  21 / 29
Parametric verification for DTPNs

Parametric reachability - a general problem

Given a property $p$, we want to find:

- the minimal time $c \in N$ at which a state satisfying $p$ can be reached

(corresponds to finding the minimal $c$ s.t. $\text{EF} \leq c p$ or $\text{EF} < c p$ holds),

Details of the verification method:
A general solution

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^c \leq p$:

1. test whether $p$ is reachable
2. if so, extract the time $x$ at which it has been reached (we know that $c \leq \lceil x \rceil$)
3. check whether there is a path of a shorter time at which $p$ is reachable
4. if such a path exists return to 2, otherwise return $\lceil x \rceil$
A general solution

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^c_p$:

1. test whether $p$ is reachable
2. if so, extract the time $x$ at which it has been reached (we know that $c \leq \lceil x \rceil$)
3. check whether there is a path of a shorter time at which $p$ is reachable
4. if such a path exists return to 2, otherwise return $\lceil x \rceil$
A general solution

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^\leq c p$:

1. test whether $p$ is reachable
2. if so, extract the time $x$ at which it has been reached (we know that $c \leq \lceil x \rceil$)
3. check whether there is a path of a shorter time at which $p$ is reachable
4. if such a path exists return to 2, otherwise return $\lceil x \rceil$
A general solution

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^c p$:

1. test whether $p$ is reachable
2. if so, extract the time $x$ at which it has been reached (we know that $c \leq \lceil x \rceil$)
3. check whether there is a path of a shorter time at which $p$ is reachable
4. if such a path exists return to 2, otherwise return $\lceil x \rceil$
A general solution

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^c p$:

1. test whether $p$ is reachable
2. if so, extract the time $x$ at which it has been reached (we know that $c \leq \lceil x \rceil$)
3. check whether there is a path of a shorter time at which $p$ is reachable
4. if such a path exists return to 2, otherwise return $\lceil x \rceil$
Solving the problem using BMC

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^{\leq c} p$:

- we run the standard reachability test to find the first time value $x$ at which $p$ can be reached
  
  we obtain a shortest path (of a length $k_0$), but not necessarily of the shortest time

- in order to test whether $p$ can be reached at the time shorter than $n$, we augment the net with an additional component and test reachability of $p \land p_{in}$

we can start with $k = k_0$

- in order to know that a state is unreachable, we need either to run proving unreachability, or to find an upper bound on the path

for certain types of nets such an upper bound can be deduced
Parametric Reachability

Solving the problem using BMC

Searching for a minimal $c \in \mathbb{N}$ s.t. $\text{EF}^{\leq c} p$:

- we run the standard reachability test to find the first time value $x$ at which $p$ can be reached.

  *we obtain a shortest path (of a length $k_0$), but not necessarily of the shortest time*

- in order to test whether $p$ can be reached at the time shorter than $n$, we augment the net with an additional component and test reachability of $p \wedge p_{in}$

- we can start with $k = k_0$

- in order to know that a state is unreachable, we need either to run proving unreachability, or to find an upper bound on the path

  *for certain types of nets such an upper bound can be deduced*
Solving the problem using BMC

Searching for a minimal \( c \in \mathbb{N} \) s.t. \( \text{EF}^{\leq c} p \):

- we run the standard reachability test to find the first time value \( x \) at which \( p \) can be reached

  *we obtain a shortest path (of a length \( k_0 \)), but not necessarily of the shortest time*

- in order to test whether \( p \) can be reached at the time shorter than \( n \), we augment the net with an additional component and test reachability of \( p \land p_{in} \)

- we can start with \( k = k_0 \)

- in order to know that a state is unreachable, we need either to run proving unreachability, or to find an upper bound on the path

  *for certain types of nets such an upper bound can be deduced*
Parametric Reachability

**Solving the problem using BMC**

Searching for a minimal $c \in \mathbb{N}$ s.t. $EF^{\leq c} p$:

- we run the standard reachability test to find the first time value $x$ at which $p$ can be reached.
  
  *we obtain a shortest path (of a length $k_0$), but not necessarily of the shortest time*

- in order to test whether $p$ can be reached at the time shorter than $n$, we augment the net with an additional component and test reachability of $p \land p_{in}$

  ![Diagram](image)

  *we can start with $k = k_0$*

- in order to know that a state is unreachable, we need either to run proving unreachability, or to find an upper bound on the path.

  *for certain types of nets such an upper bound can be deduced*
VerICS: architecture

UML
Language Translator

Estelle
Language Translator

Java
Language Translator

Promela
Language Translator

Intermediate Language

TA Translator

Parametric reachability property

PRTECTL

Time Petri Nets

Elementary Petri Nets

Reachability property

BMC4UML

BMC4TPN

BMC4EPN

Splitter

CTLK

UMC

ECTLK

BMC

TECTL

BMC4TADD

TADD

Timed Automata
### Experimental Results

EPNs: mutex of NoP processes; \( \varphi^b_1 = \forall \Theta \leq b \text{EF}(\neg p \land \text{EG} \leq \Theta c_1) \)

<table>
<thead>
<tr>
<th>formula</th>
<th>NoP</th>
<th>(k)</th>
<th>PBMC vars</th>
<th>PBMC clauses</th>
<th>MiniSAT sec</th>
<th>MB sec</th>
<th>SAT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\varphi_1^1)</td>
<td>3</td>
<td>2</td>
<td>1063</td>
<td>2920</td>
<td>0.01</td>
<td>1.3</td>
<td>NO</td>
</tr>
<tr>
<td>(\varphi_1^1)</td>
<td>3</td>
<td>3</td>
<td>1505</td>
<td>4164</td>
<td>0.01</td>
<td>1.5</td>
<td>YES</td>
</tr>
<tr>
<td>(\varphi_1^2)</td>
<td>3</td>
<td>4</td>
<td>2930</td>
<td>8144</td>
<td>0.01</td>
<td>1.5</td>
<td>NO</td>
</tr>
<tr>
<td>(\varphi_1^2)</td>
<td>3</td>
<td>5</td>
<td>3593</td>
<td>10010</td>
<td>0.01</td>
<td>1.6</td>
<td>YES</td>
</tr>
<tr>
<td>(\varphi_1^2)</td>
<td>30</td>
<td>4</td>
<td>37825</td>
<td>108371</td>
<td>0.3</td>
<td>7.4</td>
<td>NO</td>
</tr>
<tr>
<td>(\varphi_1^2)</td>
<td>30</td>
<td>5</td>
<td>46688</td>
<td>133955</td>
<td>0.4</td>
<td>8.9</td>
<td>YES</td>
</tr>
<tr>
<td>(\varphi_1^3)</td>
<td>4</td>
<td>6</td>
<td>8001</td>
<td>22378</td>
<td>0.06</td>
<td>2.5</td>
<td>NO</td>
</tr>
<tr>
<td>(\varphi_1^3)</td>
<td>4</td>
<td>7</td>
<td>9244</td>
<td>25886</td>
<td>0.05</td>
<td>2.8</td>
<td>YES</td>
</tr>
</tbody>
</table>
**DTPNs:** Fischer’s protocol of 25 processes; $\Delta = 2$, $\delta = 1$; searching for minimal $c$ s.t. $\text{EF} \leq c \cdot p$, where $p$ - violation of mutual exclusion

<table>
<thead>
<tr>
<th>$k$</th>
<th>$n$</th>
<th>variables</th>
<th>clauses</th>
<th>sec</th>
<th>MB</th>
<th>sec</th>
<th>MB</th>
<th>sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>840</td>
<td>2194</td>
<td>0.0</td>
<td>3.2</td>
<td>0.0</td>
<td>1.4</td>
<td>NO</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>16263</td>
<td>47707</td>
<td>0.5</td>
<td>5.2</td>
<td>0.1</td>
<td>4.9</td>
<td>NO</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>33835</td>
<td>99739</td>
<td>1.0</td>
<td>7.3</td>
<td>0.6</td>
<td>9.1</td>
<td>NO</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>51406</td>
<td>151699</td>
<td>1.6</td>
<td>9.6</td>
<td>1.8</td>
<td>13.8</td>
<td>NO</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>72752</td>
<td>214853</td>
<td>2.4</td>
<td>12.3</td>
<td>20.6</td>
<td>27.7</td>
<td>NO</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>92629</td>
<td>273491</td>
<td>3.0</td>
<td>14.8</td>
<td>321.4</td>
<td>200.8</td>
<td>NO</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>113292</td>
<td>334357</td>
<td>3.7</td>
<td>17.5</td>
<td>14.3</td>
<td>39.0</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>120042</td>
<td>354571</td>
<td>4.1</td>
<td>18.3</td>
<td>45.7</td>
<td>59.3</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>120054</td>
<td>354613</td>
<td>4.0</td>
<td>18.3</td>
<td>312.7</td>
<td>206.8</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>120102</td>
<td>354763</td>
<td>4.0</td>
<td>18.3</td>
<td>64.0</td>
<td>77.7</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>120054</td>
<td>354601</td>
<td>4.1</td>
<td>18.3</td>
<td>8.8</td>
<td>35.0</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>115475</td>
<td>340834</td>
<td>3.9</td>
<td>17.7</td>
<td>24.2</td>
<td>45.0</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>115481</td>
<td>340852</td>
<td>3.9</td>
<td>17.8</td>
<td>138.7</td>
<td>100.8</td>
<td>YES</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>115529</td>
<td>341008</td>
<td>3.9</td>
<td>17.7</td>
<td>2355.4</td>
<td>433.4</td>
<td>NO</td>
</tr>
</tbody>
</table>

40.1 18.3 3308.3 433.4
Final Remarks

- Parametric BMC for EPN and DTPN,
- New modules of VerICS are aimed at SAT-based parametric verification of Elementary Petri Nets, Distributed Time Petri Nets, and UML,
- Available at http://pegaz.ipipan.waw.pl/verics/