Bounded Parametric Model Checking for Petri Nets

Wojciech Penczek

ICS, Polish Academy of Sciences

Advanced Course on Petri Nets, Rostock 2010

Outline

Model Checking

Standard

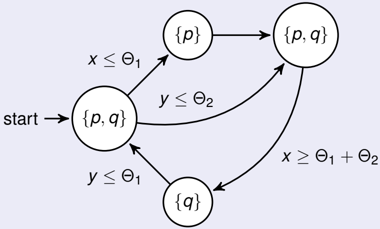
$$M \overset{?}{\models} \varphi$$

a Kripke model a modal formula

Parametric Model Checking

Parameters can appear in:

- a (timed) model[1]
- a formula[2,3]
- a model and a formula[4]



$$\forall_{\Theta \leq b} EF(\neg p \wedge EG^{\leq \Theta} c_1)$$

[1] **T. Hune, J. Romijn, M. Stoelinga, F. Vaandrager**, *Linear parametric model checking of timed automata*, TACAS'01, LNCS 2031, 2001, pp. 189–203.

[2] **V. Bruyére, E. Dall'Olio, J-F. Raskin**, *Durations and Parametric Model-Checking in Timed Automata*, ACM Transactions on Computational Logic 9(2), 2008, pp. 1–21.

[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

[4] **F. Raskin, V. Bruyère**, *Real-Time Model Checking: Parameters Everywhere*, FSTTCS'03, LNCS 2914, 2003, pp. 100–111.

Parametric Model Checking

Parameters can appear in:

- a (timed) model[1]
- a formula[2,3]
- a model and a formula[4]

$$\forall_{\Theta \leq b} EF(\neg p \wedge EG^{\leq \Theta} c_1)$$

[1] **T. Hune, J. Romijn, M. Stoelinga, F. Vaandrager**, *Linear parametric model checking of timed automata*, TACAS'01, LNCS 2031, 2001, pp. 189–203.

[2] **V. Bruyére, E. Dall'Olio, J-F. Raskin**, *Durations and Parametric Model-Checking in Timed Automata*, ACM Transactions on Computational Logic 9(2), 2008, pp. 1–21.

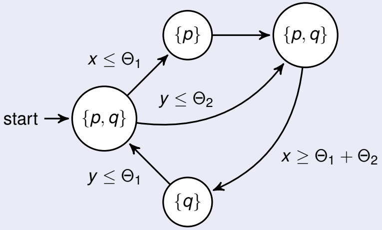
[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

[4] **F. Raskin, V. Bruyère**, *Real-Time Model Checking: Parameters Everywhere*, FSTTCS'03, LNCS 2914, 2003, pp. 100–111.

Parametric Model Checking

Parameters can appear in:

- a (timed) model[1]
- a formula[2,3]
- a model and a formula[4]



$$\forall_{\Theta \leq b} EF(\neg p \wedge EG^{\leq \Theta} c_1)$$

[1] **T. Hune, J. Romijn, M. Stoelinga, F. Vaandrager**, *Linear parametric model checking of timed automata*, TACAS'01, LNCS 2031, 2001, pp. 189–203.

[2] **V. Bruyére, E. Dall'Olio, J-F. Raskin**, *Durations and Parametric Model-Checking in Timed Automata*, ACM Transactions on Computational Logic 9(2), 2008, pp. 1–21.

[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

[4] **F. Raskin, V. Bruyère**, *Real-Time Model Checking: Parameters Everywhere*, FSTTCS'03, LNCS 2914, 2003, pp. 100–111.

Parametric Model Checking

Complexity

If parameters are in:

- a model (timed automaton), then reachability is undecidable,
- a formula, then for TECTL – 3*EXPTIME*,
- both a model and a formula, then reachability is undecidable.

Idea

Bounded Model Checking based on SAT applied to parametric model checking.

Applications

BMC for PRTCTL[3]: parameters are in formulas and parametric reachability for time Petri Nets.

[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

Parametric Model Checking

Complexity

If parameters are in:

- a model (timed automaton), then reachability is undecidable,
- a formula, then for TECTL – 3*EXPTIME*,
- both a model and a formula, then reachability is undecidable.

Idea

Bounded Model Checking based on SAT applied to parametric model checking.

Applications

BMC for PRTCTL[3]: parameters are in formulas and parametric reachability for time Petri Nets.

[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

Parametric Model Checking

Complexity

If parameters are in:

- a model (timed automaton), then reachability is undecidable,
- a formula, then for TECTL – $3EXPTIME$,
- both a model and a formula, then reachability is undecidable.

Idea

Bounded Model Checking based on SAT applied to parametric model checking.

Applications

BMC for PRTCTL[3]: parameters are in formulas and parametric reachability for time Petri Nets.

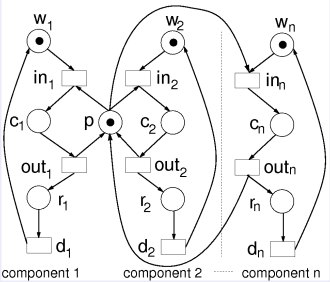
[3] **E.A. Emerson, R. Trefler**, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

Working example: mutual exclusion

Petri Net: MUTEX

Mutual exclusion:

- n processes compete for access to the shared resource
- 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 vRTCTL

- $\mathcal{PV}$ – propositional formulae, 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* $c_0, \ldots, c_n \in \mathbb{N}$.

vRTCTL syntax:

- $\mathcal{PV} \subseteq \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $\neg\alpha, \alpha \vee \beta, \alpha \wedge \beta \in \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $EX\alpha,\ EG\alpha,\ E\alpha U\beta \in \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $EG^{\leq\eta}\alpha,\ E\alpha U^{\leq\eta}\beta \in \mathrm{vRTCTL}$.

Example

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

($EF\alpha = EtrueU\alpha$ – *a derived modality*)

Syntax and semantics

Syntax of vRTCTL

- $\mathcal{PV}$ – propositional formulae, 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* $c_0, \ldots, c_n \in \mathbb{N}$*.*

vRTCTL syntax:

- $\mathcal{PV} \subseteq \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $\neg\alpha, \alpha \vee \beta, \alpha \wedge \beta \in \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $EX\alpha,\ EG\alpha,\ E\alpha U\beta \in \mathrm{vRTCTL}$,
- if $\alpha, \beta \in \mathrm{vRTCTL}$, then $EG^{\leq\eta}\alpha,\ E\alpha U^{\leq\eta}\beta \in \mathrm{vRTCTL}$.

Example

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

$$(EF\alpha = E\,true\,U\alpha \text{ – a derived modality})$$

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. $\forall_{s \in S} \exists_{s' \in S}\ s \rightarrow s'$,
- $\mathcal{L} : S \longrightarrow 2^{\mathcal{PV}}$ – a labelling function s.t. $\forall_{s \in S}\ true \in \mathcal{L}(s)$.

Parameter valuations

vRTCTL formulae are interpreted under parameter valuations:

- $\upsilon : Parameters \rightarrow \mathbb{N}$,
- υ is extended to the linear expressions η.

Example

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

$\varphi(\upsilon) = EF(\neg p \wedge EG^{\leq 2} c_1)$

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. $\forall_{s \in S} \exists_{s' \in S}\ s \rightarrow s'$,
- $\mathcal{L} : S \longrightarrow 2^{\mathcal{PV}}$ – a labelling function s.t. $\forall_{s \in S}\ true \in \mathcal{L}(s)$.

Parameter valuations

vRTCTL formulae are interpreted under parameter valuations:

- $\upsilon : Parameters \rightarrow \mathbb{N}$,
- υ is extended to the linear expressions η.

Example

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

$$\varphi(\upsilon) = EF(\neg p \wedge EG^{\leq 2} c_1)$$

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. $\forall_{s \in S} \exists_{s' \in S}\ s \rightarrow s'$,
- $\mathcal{L} : S \longrightarrow 2^{\mathcal{PV}}$ – a labelling function s.t. $\forall_{s \in S}\ true \in \mathcal{L}(s)$.

Parameter valuations

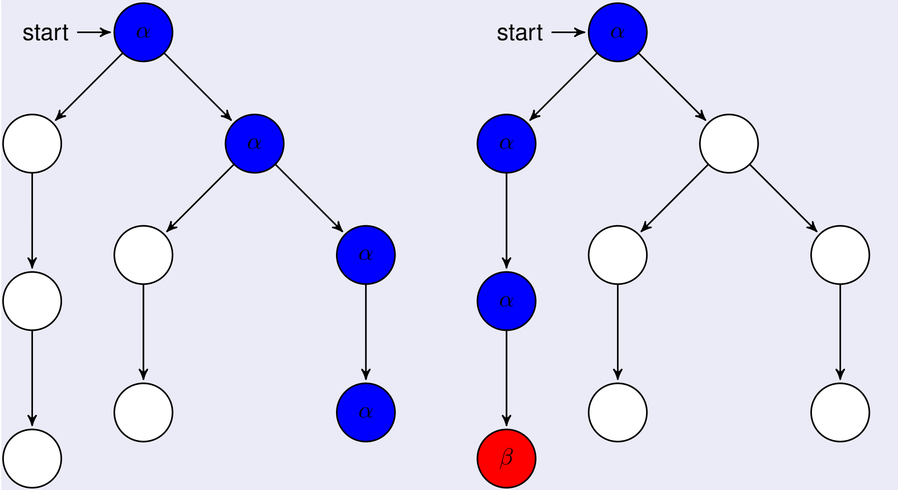
vRTCTL formulae are interpreted under parameter valuations:

- $\upsilon : Parameters \rightarrow \mathbb{N}$,
- υ is extended to the linear expressions η.

Example

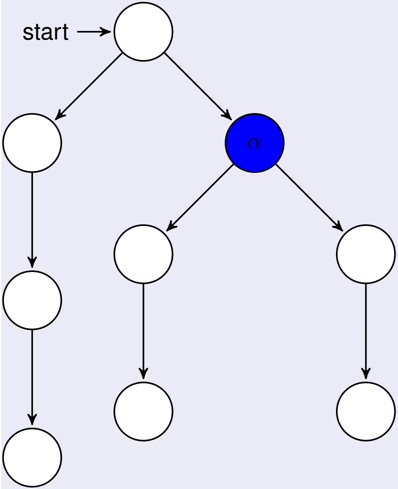
For $\varphi(\Theta) = EF(\neg p \wedge EG^{\leq \Theta} c_1)$ and υ s.t. $\upsilon(\Theta) = 2$
$\varphi(\upsilon) = EF(\neg p \wedge EG^{\leq 2} c_1)$

Syntax and semantics

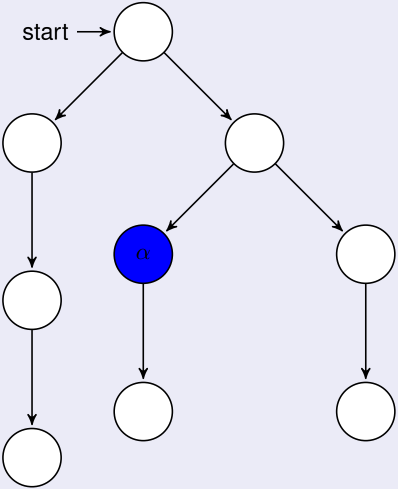


$M, start \models EG^{\leq 3}\alpha$ $\quad\quad$ $M, start \models E\alpha U^{\leq 3}\beta$

Syntax and Semantics



$M, start \models EX\alpha$



$M, start \models EF^{\leq 2}\alpha$

Syntax

Syntax of PRTCTL

- $\mathrm{vRTCTL} \subseteq \mathrm{PRTCTL}$,
- if $\alpha(\Theta) \in \mathrm{vRTCTL} \cup \mathrm{PRTCTL}$, then $\forall_\Theta \alpha(\Theta), \exists_\Theta \alpha(\Theta), \forall_{\Theta \leq a} \alpha(\Theta), \exists_{\Theta \leq a} \alpha(\Theta) \in \mathrm{PRTCTL}$ for $a \in \mathbb{N}$.

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

Example: $\varphi_1^3 = \forall_{\Theta \leq 3} EF(\neg p \wedge EG^{\leq \Theta} c_1)$

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

Syntax

Syntax of PRTCTL

- vRTCTL $\subseteq$ PRTCTL,
- if $\alpha(\Theta) \in$ vRTCTL $\cup$ 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 α.

Example: $\varphi_1^3 = \forall_{\Theta \leq 3} EF(\neg p \wedge EG^{\leq \Theta} c_1)$

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

Semantics

Semantics* of PRTCTL (the closed formulae)

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

*) The red part follows from a theorem.

Example: $M, s \models \varphi_1^3$ iff $\bigwedge_{i_\Theta \leq 3} M, s \models EF(\neg p \wedge EG^{\leq i_\Theta} c_1)$

Semantics

Semantics* of PRTCTL (the closed formulae)

- $M, s \models \forall_\Theta \alpha(\Theta)$ iff $\bigwedge_{0 \le i_\Theta \le |M|} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \forall_{\Theta \le a} \alpha(\Theta)$ iff $\bigwedge_{0 \le i_\Theta \le a} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \exists_\Theta \alpha(\Theta)$ iff $\bigvee_{0 \le i_\Theta \le |M|} M, s \models \alpha(\Theta \leftarrow i_\Theta)$,
- $M, s \models \exists_{\Theta \le a} \alpha(\Theta)$ iff $\bigvee_{0 \le i_\Theta \le a} M, s \models \alpha(\Theta \leftarrow i_\Theta)$.

*) The red part follows from a theorem.

Example: $M, s \models \varphi_1^3$ iff $\bigwedge_{i_\Theta \le 3} M, s \models EF(\neg p \wedge EG^{\le i_\Theta} c_1)$

Example of a PRTCTL formula

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

expresses much more than the corresponding CTL formula

$$[AG(request \Rightarrow AF receive) \Rightarrow AG(request \Rightarrow AF grant)]$$

Complexity of model checking

For CTL, vRTCTL, and PRTCTL

- CTL and RTCTL 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 φ.

E.A. Emerson, R. Trefler, *Parametric quantitative temporal reasoning*, LICS'99, 1999, pp. 336–343.

Existential fragments

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

Example: $\varphi_1^4 = \forall_{\Theta \leq 4} EF(\neg p \wedge EG^{\leq \Theta} c_1)$

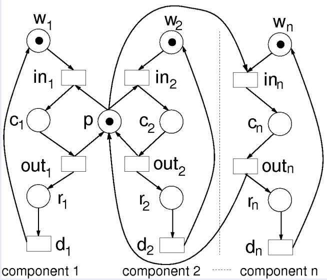
Existential fragments

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

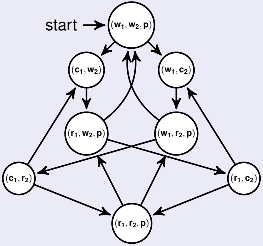
Example: $\varphi_1^4 = \forall_{\Theta \leq 4} EF(\neg p \wedge EG^{\leq \Theta} c_1)$

Syntax and semantics – back to MUTEX

Petri Net for MUTEX



The marking graph



Let $\varphi_1^b = \forall_{\Theta \leq b} EF(\neg p \wedge EG^{\leq \Theta} c_1)$.

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

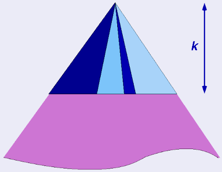
"There exists a future state, such that the resource is taken and the first process stays in the critical section for any time value bounded by b"

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.

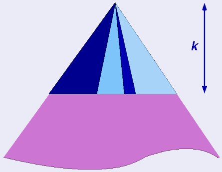


Bounded semantics

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.

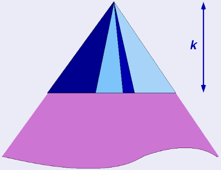


Bounded semantics

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.

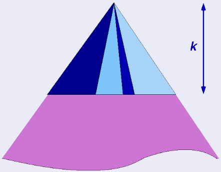


Bounded semantics

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.

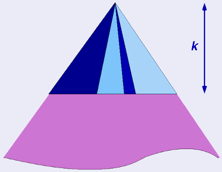


Bounded semantics

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.

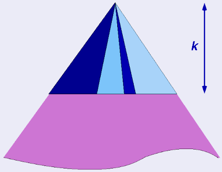


Bounded semantics

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 $s_i \to s_{i+1}$.
- $M_k = (Path_k, \mathcal{L})$ is called the *k-model*.
- If an existential formula φ holds in M_k, then φ holds in M.
- The problem $M_k \models \varphi$ is translated to checking satisfiability of the propositional formula $[M_k] \wedge [\varphi]$ using a SAT-solver.



SAT solvers

SAT

- Problem: is a propositional formula satisfiable?
- Theoretical complexity: NP-complete (Cook, 1971)
- Practical and efficient realizations of SAT solvers: only in the last decade
- A general idea: search efficiently for a satisfying assignment

Details

- Efficient data representation
- Heuristics for deducing and learning information
- Frequently efficient in practice
- CNF: conjunctive normal form, conjunction of disjunctions of literals

$$\varphi = (a \vee b \vee \neg c) \wedge (\neg c) \wedge (a \vee \neg b)$$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \to \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff
  $\exists_{\pi_k \in Path_k}\left(\pi_k(0) = s \wedge \left[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)\right.\right.$
  $\left.\left.\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\right]\right)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq\eta}\beta)$ iff
  $\exists_{\pi_k \in Path_k}\left(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k,\upsilon(\eta))}\left[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\right]\right)$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \rightarrow \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff $\exists_{\pi_k \in Path_k}\left(\pi_k(0) = s \wedge \left[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha) \vee ((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\right]\right)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq\eta}\beta)$ iff $\exists_{\pi_k \in Path_k}\left(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k,\upsilon(\eta))}\left[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\right]\right)$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \to \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff $\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha) \vee ((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\big)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq\eta}\beta)$ iff $\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k,\upsilon(\eta))}\big[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\big]\big)$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \to \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff $\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$ $\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\big)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq\eta}\beta)$ iff $\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k,\upsilon(\eta))}\big[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\big]\big)$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \to \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff
  $\exists_{\pi_k \in Path_k}\Big(\pi_k(0) = s \wedge \Big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$
  $\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\Big]\Big)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq\eta}\beta)$ iff
  $\exists_{\pi_k \in Path_k}\Big(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k,\upsilon(\eta))}\Big[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\Big]\Big)$

Bounded semantics of vRTECTL

- $\alpha(\upsilon)$ – the formula obtained by substituting the parameters in α according to υ,
- $M_k, s \models_\upsilon \alpha$ iff $M_k, s \models \alpha(\upsilon)$.

Loop predicate – to represent infinite paths in k–model:

- $loop(\pi_k) = true$ iff $\pi_k(k) \rightarrow \pi_k(l)$ for some $l \leq k$

- $M_k, s \models_\upsilon p$ iff $p \in \mathcal{L}(s)$, and $s \models_\upsilon \neg p$ iff $p \notin \mathcal{L}(s)$
- $M_k, s \models_\upsilon \alpha \wedge \beta$ iff $s \models_\upsilon \alpha \wedge s \models_\upsilon \beta$, and $s \models_\upsilon \alpha \vee \beta$ iff $s \models_\upsilon \alpha \vee s \models_\upsilon \beta$
- $M_k, s \models_\upsilon EX\alpha$ iff $\exists_{\pi_k \in Path_k}(\pi_k(0) = s \wedge \pi_k(1) \models_\upsilon \alpha)$
- $M_k, s \models_\upsilon EG^{\leq \eta}\alpha$ iff
$\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$
$\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\big)$
- $M_k, s \models_\upsilon E(\alpha U^{\leq \eta}\beta)$ iff
$\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \exists_{0 \leq i \leq min(k, \upsilon(\eta))}\big[\pi_k(i) \models_\upsilon \beta \wedge \bigwedge_{0 \leq j < i} \pi_k(i) \models_\upsilon \alpha\big]\big)$

Bounded semantics of vRTECTL

$EG^{\leq\eta}$ – illustration

$M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff
$\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0\leq i\leq\upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$
$\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0\leq i\leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\big)$

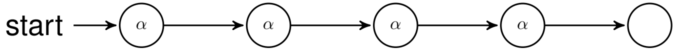
$\upsilon(\eta) \leq k$: the whole prefix of length $\upsilon(\eta)$ is contained in the k-path

$\upsilon(\eta) > k$: a loop detection

Bounded semantics of vRTECTL

$EG^{\leq \eta}$ – illustration

$M_k, s \models_\upsilon EG^{\leq \eta}\alpha$ iff
$\exists_{\pi_k \in Path_k} \Big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0 \leq i \leq \upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$
$\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0 \leq i \leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\Big)$



$\upsilon(\eta) \leq k$: the whole prefix of length $\upsilon(\eta)$ is contained in the k-path

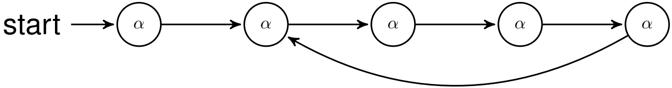
$\upsilon(\eta) > k$: a loop detection

Bounded semantics of vRTECTL

$EG^{\leq\eta}$ – illustration

$M_k, s \models_\upsilon EG^{\leq\eta}\alpha$ iff
$\exists_{\pi_k \in Path_k}\big(\pi_k(0) = s \wedge \big[((\upsilon(\eta) \leq k) \wedge \bigwedge_{0\leq i\leq\upsilon(\eta)} \pi_k(i) \models_\upsilon \alpha)$
$\vee((\upsilon(\eta) > k) \wedge \bigwedge_{0\leq i\leq k} \pi_k(i) \models_\upsilon \alpha \wedge loop(\pi_k))\big]\big)$

$\upsilon(\eta) \leq k$: the whole prefix of length $\upsilon(\eta)$ is contained in the k-path



$\upsilon(\eta) > k$: a loop detection

Bounded semantics of PRTECTL

The bounded quantifiers are replaced by disjunctions and/or conjunctions.

- $M_k, s \models \forall_{\Theta \leq a} \alpha(\Theta)$ iff $\bigwedge_{0 \leq i_\Theta \leq a} M_k, s \models \alpha(\Theta \leftarrow i_\Theta)$
- $M_k, s \models \exists_{\Theta \leq a} \alpha(\Theta)$ iff $\bigvee_{0 \leq i_\Theta \leq min(a,k)} M_k, s \models \alpha(\Theta \leftarrow i_\Theta)$

Example:

$M_3, s \models \exists_{\Theta_1 \leq 4} \forall_{\Theta_2 \leq 5} EF^{\leq \Theta_1}(\neg p \wedge EG^{\leq \Theta_2} c_1)$

iff

$\bigvee_{i_{\Theta_1} \leq 3} \bigwedge_{i_{\Theta_2} \leq 5} M_3, s \models EF^{\leq i_{\Theta_1}}(\neg p \wedge EG^{\leq i_{\Theta_2}} c_1)$

Bounded semantics of PRTECTL

The bounded quantifiers are replaced by disjunctions and/or conjunctions.

- $M_k, s \models \forall_{\Theta \leq a} \alpha(\Theta)$ iff $\bigwedge_{0 \leq i_\Theta \leq a} M_k, s \models \alpha(\Theta \leftarrow i_\Theta)$
- $M_k, s \models \exists_{\Theta \leq a} \alpha(\Theta)$ iff $\bigvee_{0 \leq i_\Theta \leq min(a,k)} M_k, s \models \alpha(\Theta \leftarrow i_\Theta)$

Example:

$$M_3, s \models \exists_{\Theta_1 \leq 4} \forall_{\Theta_2 \leq 5} EF^{\leq \Theta_1}(\neg p \wedge EG^{\leq \Theta_2} c_1)$$

iff

$$\bigvee_{i_{\Theta_1} \leq 3} \bigwedge_{i_{\Theta_2} \leq 5} M_3, s \models EF^{\leq i_{\Theta_1}}(\neg p \wedge EG^{\leq i_{\Theta_2}} c_1)$$

Translation to boolean formulae

Encoding M_k

- The states are encoded as valuations of boolean vectors $w_{i,j}$.
  - $(w_{0,j}, w_{1,j}, \ldots, w_{k,j})$ – the jth symbolic k–path
- The other elements of M_k:
  - $p(w)$, where $p \in \mathcal{PV}$ – encoding of the labelling function,
  - $T(w, w')$ – encoding of the transition relation,
  - $I_s(w)$ – encoding of the state s,
  - $H(w, w')$ – encoding of the equality of two states encodings,
  - $L_k(j) \stackrel{def}{=} \bigvee_{i=0}^{k} T(w_{k,j}, w_{i,j})$ – encoding of a loop.

Encoding of the k-paths

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

where A – a set of the path indexes determined by the function[5] F_k.

[5] **W. Penczek, B. Woźna, A. Zbrzezny**, Bounded Model Checking for the Universal Fragment of CTL, *Fundamenta Informaticae, vol. 51(1-2), 2002, pp. 135–156.*

Translation to boolean formulae

Encoding M_k

- The states are encoded as valuations of boolean vectors $w_{i,j}$.
  - $(w_{0,j}, w_{1,j}, \ldots, w_{k,j})$ – the jth symbolic k–path
- The other elements of M_k:
  - $p(w)$, where $p \in \mathcal{PV}$ – encoding of the labelling function,
  - $T(w, w')$ – encoding of the transition relation,
  - $I_s(w)$ – encoding of the state s,
  - $H(w, w')$ – encoding of the equality of two states encodings,
  - $L_k(j) \stackrel{def}{=} \bigvee_{i=0}^{k} T(w_{k,j}, w_{i,j})$ – encoding of a loop.

Encoding of the k-paths

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

where A – a set of the path indexes determined by the function[5] F_k.

[5] **W. Penczek, B. Woźna, A. Zbrzezny**, Bounded Model Checking for the Universal Fragment of CTL, *Fundamenta Informaticae, vol. 51(1-2), 2002, pp. 135–156.*

Translation to boolean formulae

Encoding M_k

- The states are encoded as valuations of boolean vectors $w_{i,j}$.
  - $(w_{0,j}, w_{1,j}, \ldots, w_{k,j})$ – the jth symbolic k–path
- The other elements of M_k:
  - $p(w)$, where $p \in \mathcal{PV}$ – encoding of the labelling function,
  - $T(w, w')$ – encoding of the transition relation,
  - $I_s(w)$ – encoding of the state s,
  - $H(w, w')$ – encoding of the equality of two states encodings,
  - $L_k(j) \stackrel{def}{=} \bigvee_{i=0}^{k} T(w_{k,j}, w_{i,j})$ – encoding of a loop.

Encoding of the k-paths

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

where A – a set of the path indexes determined by the function[5] F_k.

[5] **W. Penczek, B. Woźna, A. Zbrzezny**, Bounded Model Checking for the Universal Fragment of CTL, *Fundamenta Informaticae, vol. 51(1-2), 2002, pp. 135–156.*

Translation to boolean formulae

Encoding M_k

- The states are encoded as valuations of boolean vectors $w_{i,j}$.
  - $(w_{0,j}, w_{1,j}, \ldots, w_{k,j})$ – the jth symbolic k–path
- The other elements of M_k:
  - $p(w)$, where $p \in \mathcal{PV}$ – encoding of the labelling function,
  - $T(w, w')$ – encoding of the transition relation,
  - $I_s(w)$ – encoding of the state s,
  - $H(w, w')$ – encoding of the equality of two states encodings,
  - $L_k(j) \stackrel{def}{=} \bigvee_{i=0}^{k} T(w_{k,j}, w_{i,j})$ – encoding of a loop.

Encoding of the k-paths

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

where A – a set of the path indexes determined by the function[5] F_k.

[5] **W. Penczek, B. Woźna, A. Zbrzezny**, Bounded Model Checking for the Universal Fragment of CTL, *Fundamenta Informaticae, vol. 51(1-2), 2002, pp. 135–156.*

Translation to boolean formulae

Encoding M_k

- The states are encoded as valuations of boolean vectors $w_{i,j}$.
  - $(w_{0,j}, w_{1,j}, \ldots, w_{k,j})$ – the jth symbolic k–path
- The other elements of M_k:
  - $p(w)$, where $p \in \mathcal{PV}$ – encoding of the labelling function,
  - $T(w, w')$ – encoding of the transition relation,
  - $I_s(w)$ – encoding of the state s,
  - $H(w, w')$ – encoding of the equality of two states encodings,
  - $L_k(j) \stackrel{def}{=} \bigvee_{i=0}^{k} T(w_{k,j}, w_{i,j})$ – encoding of a loop.

Encoding of the k-paths

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

where A – a set of the path indexes determined by the function[5] F_k.

[5] **W. Penczek, B. Woźna, A. Zbrzezny**, Bounded Model Checking for the Universal Fragment of CTL, *Fundamenta Informaticae, vol. 51(1-2), 2002, pp. 135–156.*

Encoding formulae

Example of the translation[6] from vRETCTL to SAT:

if $\upsilon(\eta) > k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge L_k(min(A)) \wedge \bigwedge_{j=0}^{k} \left[\alpha\right]_k^{[j,min(A),h_G(A,k)(j),\upsilon]}$$

if $\upsilon(\eta) \leq k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge \bigwedge_{j=0}^{\upsilon(\eta)} \left[\alpha\right]_k^{[j,min(A),h_G(A,\upsilon(\eta))(j),\upsilon]}$$

Encoding $M_k, s \models \alpha$

$$\left[M\right]_k^{\alpha} := \left[M\right]_k^{F_k(\alpha)} \wedge I_s(w_{0,0}) \wedge \left[\alpha\right]_k^{[0,0,F_k(\alpha)]}$$

Theorem

The encoding is correct.

[6] **A. Zbrzezny**, *Improving the Translation from ECTL to SAT*, *Fundamenta Informaticae, vol. 85(1-4), 2008, pp. 513–531.*

Encoding formulae

Example of the translation[6] from vRETCTL to SAT:

if $\upsilon(\eta) > k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge L_k(min(A)) \wedge \bigwedge_{j=0}^{k} \left[\alpha\right]_k^{[j,min(A),h_G(A,k)(j),\upsilon]}$$

if $\upsilon(\eta) \leq k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge \bigwedge_{j=0}^{\upsilon(\eta)} \left[\alpha\right]_k^{[j,min(A),h_G(A,\upsilon(\eta))(j),\upsilon]}$$

Encoding $M_k, s \models \alpha$

$$\left[M\right]_k^{\alpha} := \left[M\right]_k^{F_k(\alpha)} \wedge I_s(w_{0,0}) \wedge \left[\alpha\right]_k^{[0,0,F_k(\alpha)]}$$

Theorem

The encoding is correct.

[6] **A. Zbrzezny**, *Improving the Translation from ECTL to SAT*, *Fundamenta Informaticae, vol. 85(1-4), 2008, pp. 513–531.*

Encoding formulae

Example of the translation[6] from vRETCTL to SAT:

if $\upsilon(\eta) > k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge L_k(min(A)) \wedge \bigwedge_{j=0}^{k} [\alpha]_k^{[j,min(A),h_G(A,k)(j),\upsilon]}$$

if $\upsilon(\eta) \leq k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge \bigwedge_{j=0}^{\upsilon(\eta)} [\alpha]_k^{[j,min(A),h_G(A,\upsilon(\eta))(j),\upsilon]}$$

Encoding $M_k, s \models \alpha$

$$\left[M\right]_k^{\alpha} := \left[M\right]_k^{F_k(\alpha)} \wedge I_s(w_{0,0}) \wedge [\alpha]_k^{[0,0,F_k(\alpha)]}$$

Theorem

The encoding is correct.

[6] **A. Zbrzezny**, *Improving the Translation from ECTL to SAT*, *Fundamenta Informaticae, vol. 85(1-4), 2008, pp. 513–531.*

Encoding formulae

Example of the translation[6] from vRETCTL to SAT:

if $\upsilon(\eta) > k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge L_k(min(A)) \wedge \bigwedge_{j=0}^{k} \left[\alpha\right]_k^{[j,min(A),h_G(A,k)(j),\upsilon]}$$

if $\upsilon(\eta) \leq k$:

$$\left[EG^{\leq\eta}\alpha\right]^{[m,n,A,\upsilon]} := H(w_{m,n}, w_{0,min(A)}) \wedge \bigwedge_{j=0}^{\upsilon(\eta)} \left[\alpha\right]_k^{[j,min(A),h_G(A,\upsilon(\eta))(j),\upsilon]}$$

Encoding $M_k, s \models \alpha$

$$\left[M\right]_k^{\alpha} := \left[M\right]_k^{F_k(\alpha)} \wedge I_s(w_{0,0}) \wedge \left[\alpha\right]_k^{[0,0,F_k(\alpha)]}$$

Theorem

The encoding is correct.

[6] **A. Zbrzezny**, *Improving the Translation from ECTL to SAT*, *Fundamenta Informaticae, vol. 85(1-4), 2008, pp. 513–531.*

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,
- $Eft : T \to \mathbb{N}$, $Lft : T \to \mathbb{N} \cup \{\infty\}$ - 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*

Distributed Time Petri Nets

Time Petri Nets

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

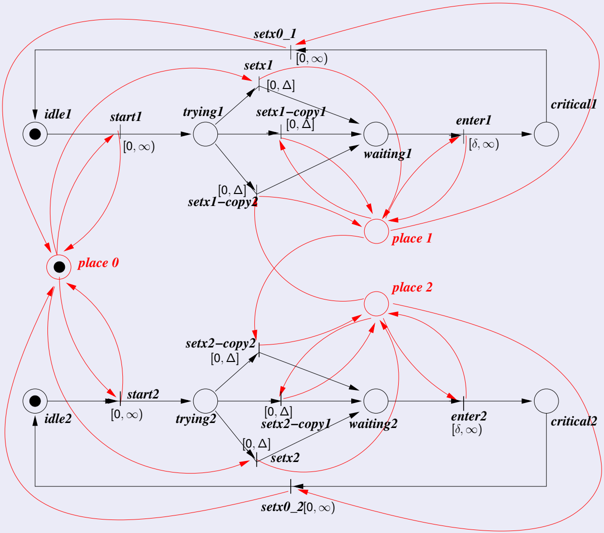
- P, T, F, m_0 - like before,
- $Eft : T \to \mathbb{N}$, $Lft : T \to \mathbb{N} \cup \{\infty\}$ - earliest and latest firing times of transitions ($Eft(t) \leq Lft(t)$ for each $t \in T$)

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



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. $\mathrm{EF}^{\leq c}\mathrm{p}$ or $\mathrm{EF}^{<c}\mathrm{p}$ holds),
- the minimal time $c \in \mathbb{N}$ after which no state satisfying p can be reached

  (corresponds to finding the maximal c s.t. $\mathrm{EF}^{\geq c}\mathrm{p}$ or $\mathrm{EF}^{>c}\mathrm{p}$ holds).

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. $\mathrm{EF}^{\leq c}\mathrm{p}$ or $\mathrm{EF}^{< c}\mathrm{p}$ holds),
- the minimal time $c \in \mathbb{N}$ after which no state satisfying p can be reached

  (corresponds to finding the maximal c s.t. $\mathrm{EF}^{\geq c}\mathrm{p}$ or $\mathrm{EF}^{> c}\mathrm{p}$ holds).

Parametric reachability

A general solution

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

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 $\lceil x \rceil$

Parametric reachability

A general solution

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

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 $\lceil x \rceil$

Parametric reachability

A general solution

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

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 $\lceil x \rceil$

Parametric reachability

A general solution

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

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 $\lceil x \rceil$

Parametric reachability

A general solution

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

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 $\lceil x \rceil$

Parametric Reachability

Solving the problem using BMC

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

- 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

Details of the verification method:

W.Penczek, A.Półrola, A.Zbrzezny: *"SAT-Based (Parametric) Reachability for Distributed Time Petri Nets"*; Proc. of PNSE'09

Parametric Reachability

Solving the problem using BMC

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

- 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

Details of the verification method:

W.Penczek, A.Półrola, A.Zbrzezny: "SAT-Based (Parametric) Reachability for Distributed Time Petri Nets"; Proc. of PNSE'09

Parametric Reachability

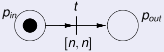
Solving the problem using BMC

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

- 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

Details of the verification method:

W.Penczek, A.Półrola, A.Zbrzezny: *"SAT-Based (Parametric) Reachability for Distributed Time Petri Nets"*; Proc. of PNSE'09

Parametric Reachability

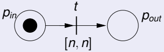
Solving the problem using BMC

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

- 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 $k = k_0$

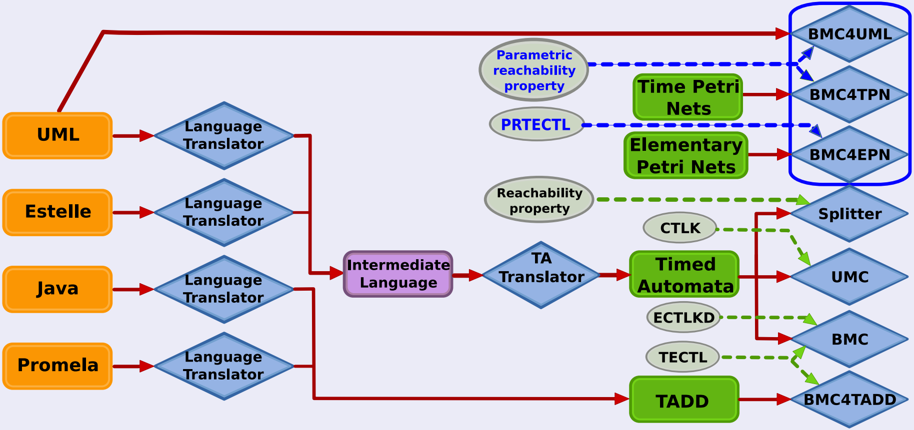
- in order to know that a state is unreachable, we need either to run proving unreachability, or to find an upper bound on the path

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

Details of the verification method:

W.Penczek, A.Półrola, A.Zbrzezny: "SAT-Based (Parametric) Reachability for Distributed Time Petri Nets"; Proc. of PNSE'09

VerICS: architecture



Parametric verification: input formalisms

Input formalisms

- Elementary Net Systems (Elementary Petri Nets),
- (Distributed) Time Petri Nets,
- A subset of UML.

Experimental Results

EPNs: mutex of NoP processes; $\varphi_1^b = \forall_{\Theta \le b} EF(\neg p \wedge EG^{\le \Theta} c_1)$

formula	NoP	k	PBMC				MiniSAT	SAT?
			vars	clauses	sec	MB	sec	
φ_1^1	3	2	1063	2920	0.01	1.3	0.003	NO
φ_1^1	3	3	1505	4164	0.01	1.5	0.008	YES
φ_1^2	3	4	2930	8144	0.01	1.5	0.01	NO
φ_1^2	3	5	3593	10010	0.01	1.6	0.03	YES
φ_1^2	30	4	37825	108371	0.3	7.4	0.2	NO
φ_1^2	30	5	46688	133955	0.4	8.9	0.52	YES
φ_1^3	4	6	8001	22378	0.06	2.5	0.04	NO
φ_1^3	4	7	9244	25886	0.05	2.8	0.05	YES

DTPNs: Fischer's protocol of 25 processes; $\Delta = 2$, $\delta = 1$; searching for minimal c s.t. $\mathbf{EF}^{\leq c}\mathbf{p}$, where p - violation of mutual exclusion

		tpnBMC				RSat		
k	**n**	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**	**sat**
0	-	840	2194	0.0	3.2	0.0	1.4	NO
2	-	16263	47707	0.5	5.2	0.1	4.9	NO
4	-	33835	99739	1.0	7.3	0.6	9.1	NO
6	-	51406	151699	1.6	9.6	1.8	13.8	NO
8	-	72752	214853	2.4	12.3	20.6	27.7	NO
10	-	92629	273491	3.0	14.8	321.4	200.8	NO
12	-	113292	334357	3.7	17.5	14.3	39.0	YES
12	7	120042	354571	4.1	18.3	45.7	59.3	YES
12	6	120054	354613	4.0	18.3	312.7	206.8	YES
12	5	120102	354763	4.0	18.3	64.0	77.7	YES
12	4	120054	354601	4.1	18.3	8.8	35.0	YES
12	3	115475	340834	3.9	17.7	24.2	45.0	YES
12	2	115481	340852	3.9	17.8	138.7	100.8	YES
12	1	115529	341008	3.9	17.7	2355.4	433.4	NO
				40.1	18.3	3308.3	433.4	

Final Remarks and Next Lecture

Final Remarks

- VerICS - a model checker for high-level languages, real-time, and multi-agent systems,
- New modules are aimed at SAT-based parametric verification of Elementary Petri Nets, Distributed Time Petri Nets, and UML,
- Avaialable at http://pegaz.ipipan.waw.pl/verics/

Next Lecture

Verification of Distributed Systems with the toolkit VerICS

The End

Thank You