



Verifying MAS using the model checker VerICS

WOJCIECH PENCZEK

ICS PAS, Warsaw, Poland

Many thanks to members of the [Verics](#) group:

M. Kacprzak, T. Lasica, A. Półrola, M. Szreter, B. Woźna, A.Zbrzezny



Talk Overview

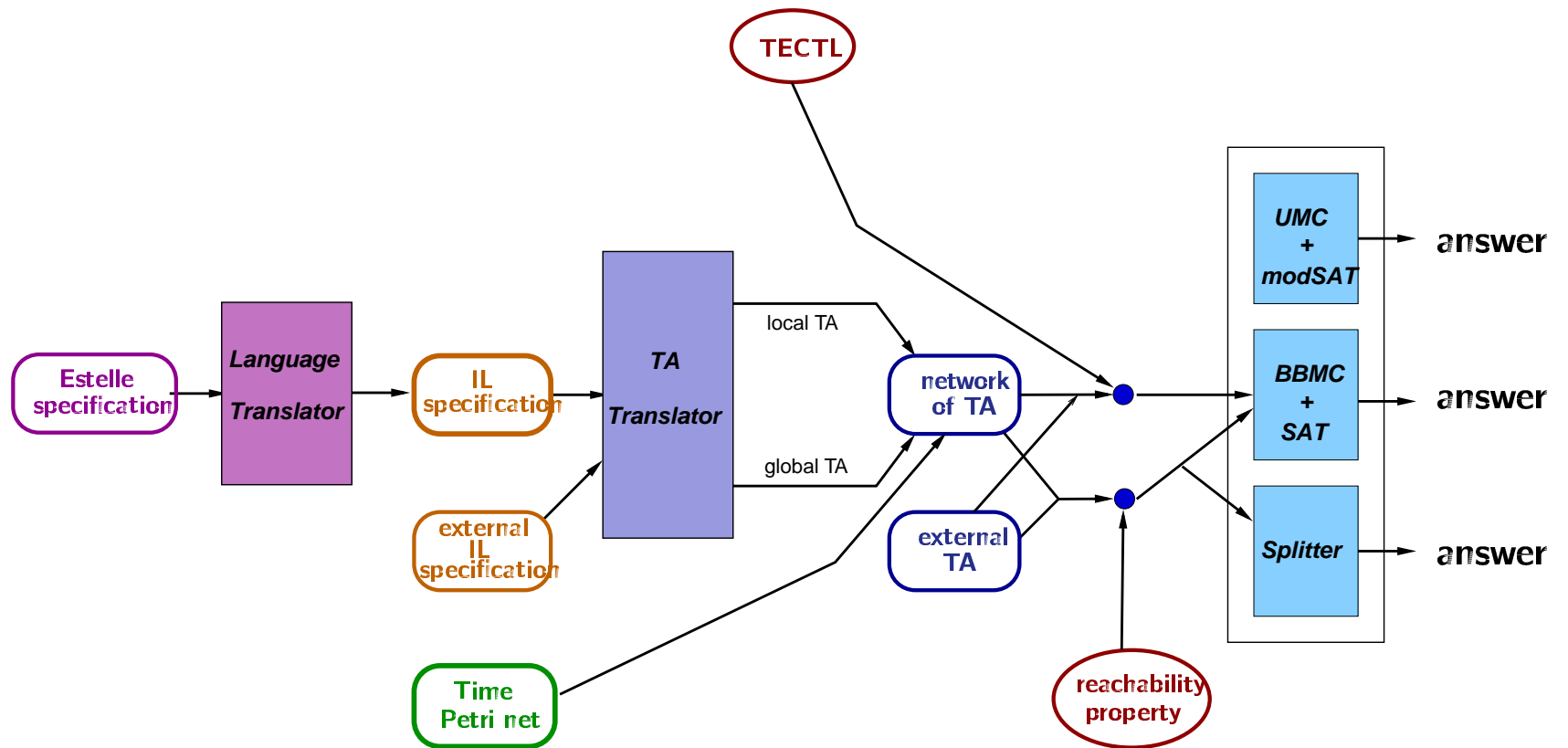
- ✧ Related work,
- ✧ VerICS architecture,
- ✧ Efficient SAT algorithms and SAT-solvers,
- ✧ SAT-based symbolic approaches:
 - ✧ **Bounded Model Checking**,
 - ✧ **Unbounded Model Checking**,
- ✧ Experimental results,
- ✧ Tool Demo.



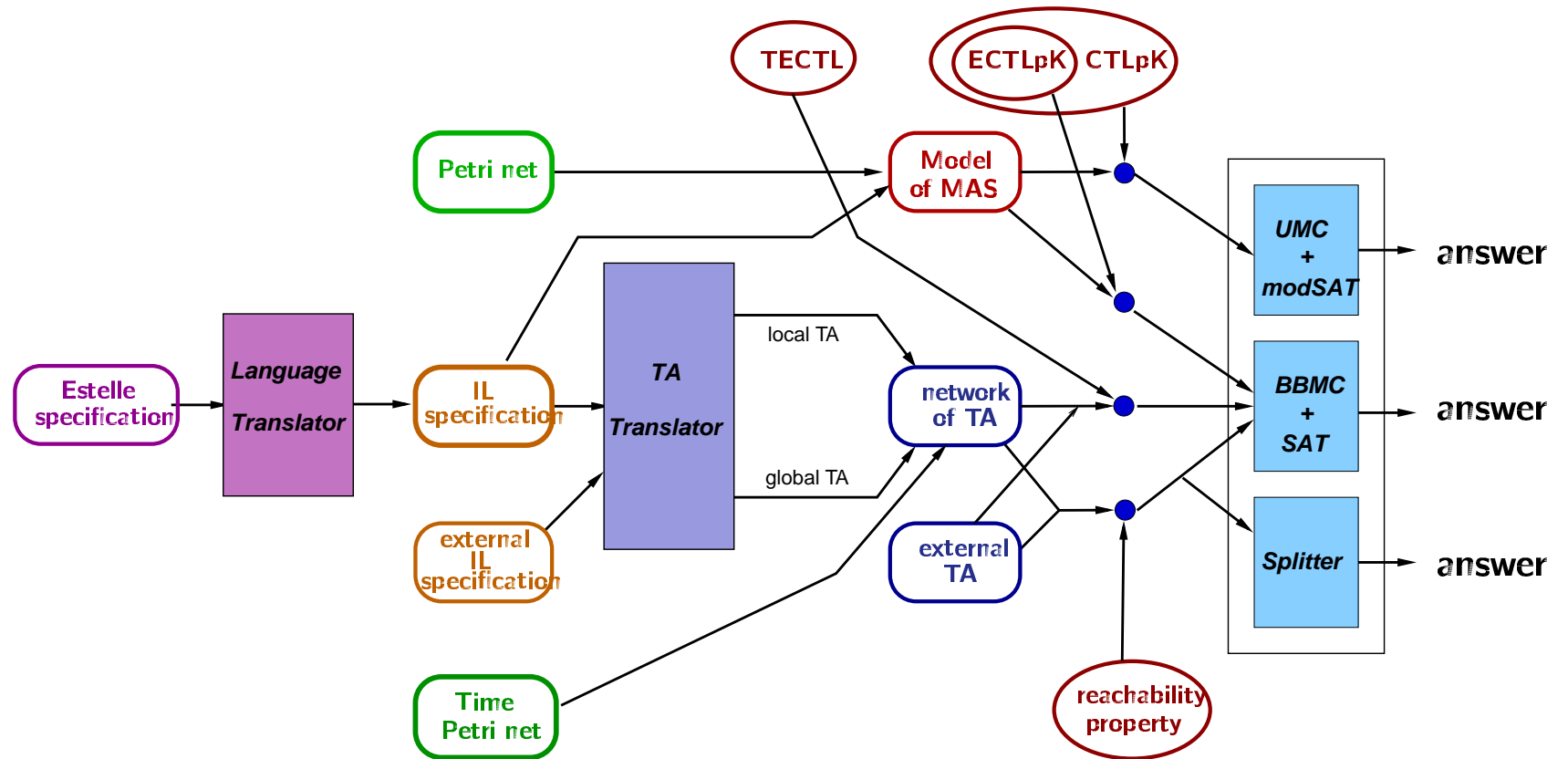
Related Work

- ✦ R. van der Meyden and H. Shilov. "Model checking knowledge and time in systems with perfect knowledge", FST&TCS'99,
- ✦ R. van der Meyden and K. Su. "Symbolic model checking the knowledge of the dining cryptographers", 2002,
- ✦ W. van der Hoek and M. Wooldridge. "Model Checking Knowledge and Time", SPIN'02,
- ✦ F. Raimondi, A. Lomuscio. Papers on BDD-based model checker for interpreted systems, ELNCS'03, AAMAS'04
- ✦ M. Wooldridge and M. Fisher and M. P. Huget and S. Parsons. "Model Checking Multiagent Systems with MABLE", (AAMAS'02),
- ✦ R. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge, "Model checking AgentSpeak", AAMAS'03,
- ✦ M. Kacprzak, A. Lomuscio, W. Penczek. Papers on BMC and UMC for interpreted systems, AAMAS'03-04,

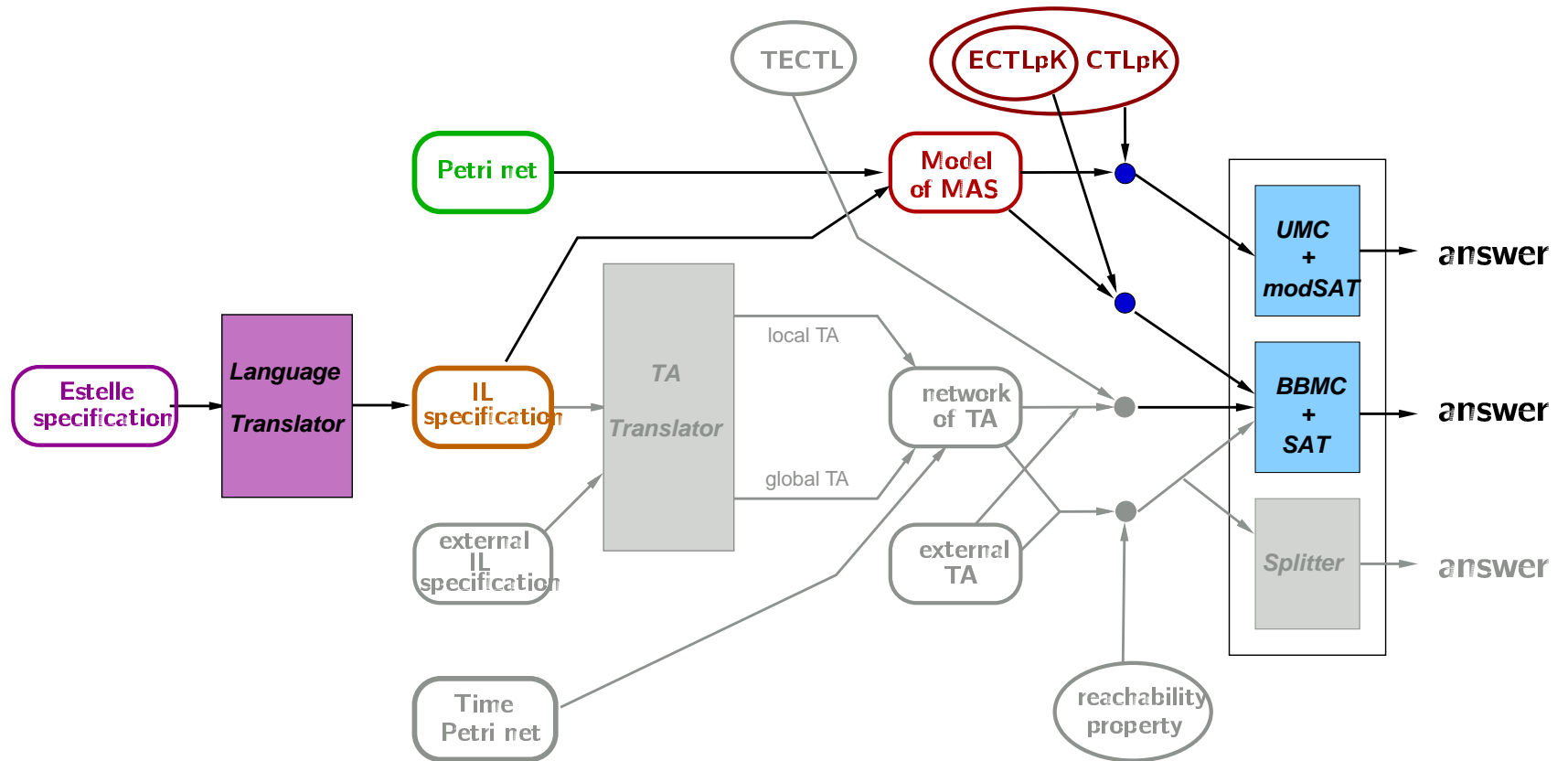
Verics architecture - timed systems



Verics full architecture



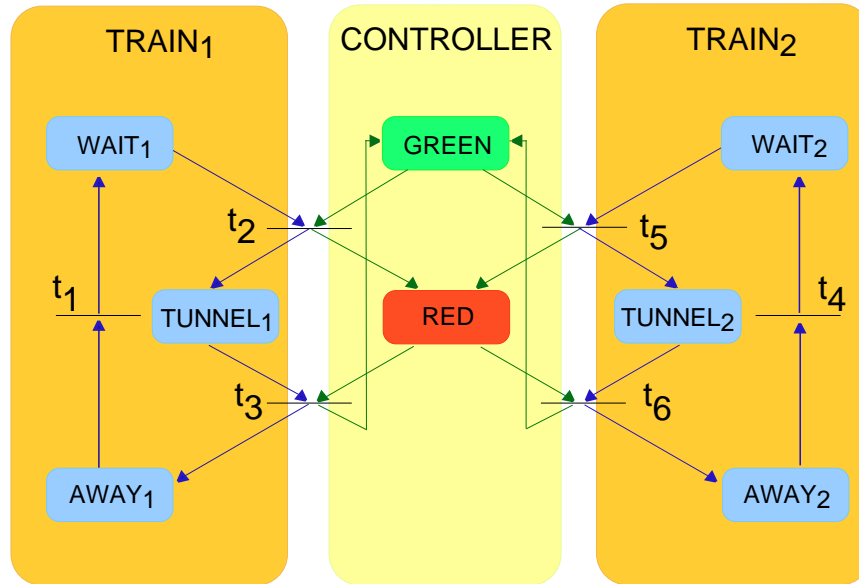
Verics architecture - MAS



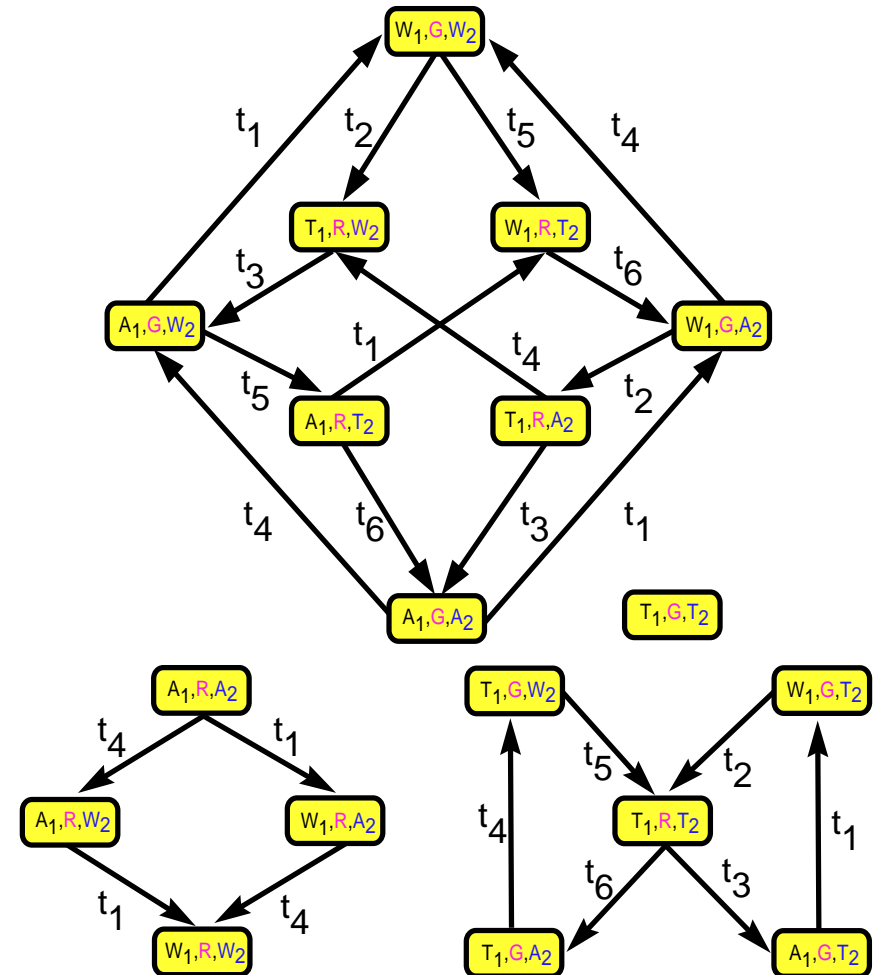
Assume the knowledge of:

- ✦ Idea of model checking,
- ✦ Interpreted systems, epistemic Kripke models,
- ✦ The logic **CTLpK**,
- ✦ Subsets of **CTLpK** : **ACTLpK**, **ECTLpK**,
- ✦ Quantified Boolean Formulas (QBF).

Interpreted Systems - Example of Trains from [M. Wooldridge]

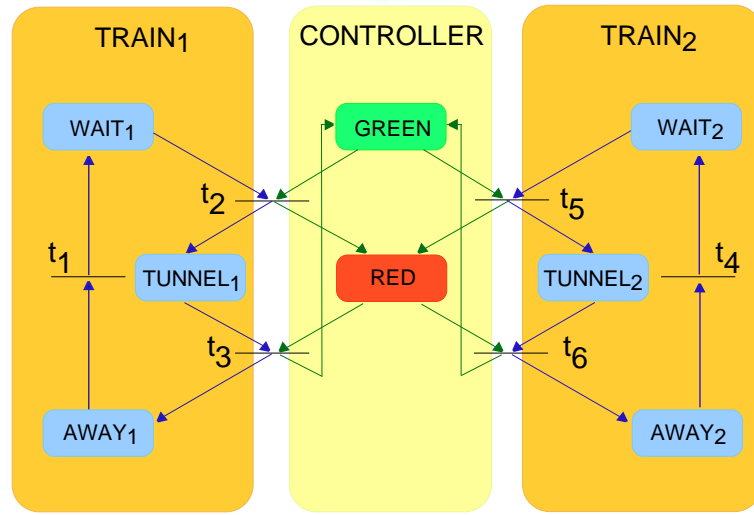


Local transition structures:
Two trains and a tunnel

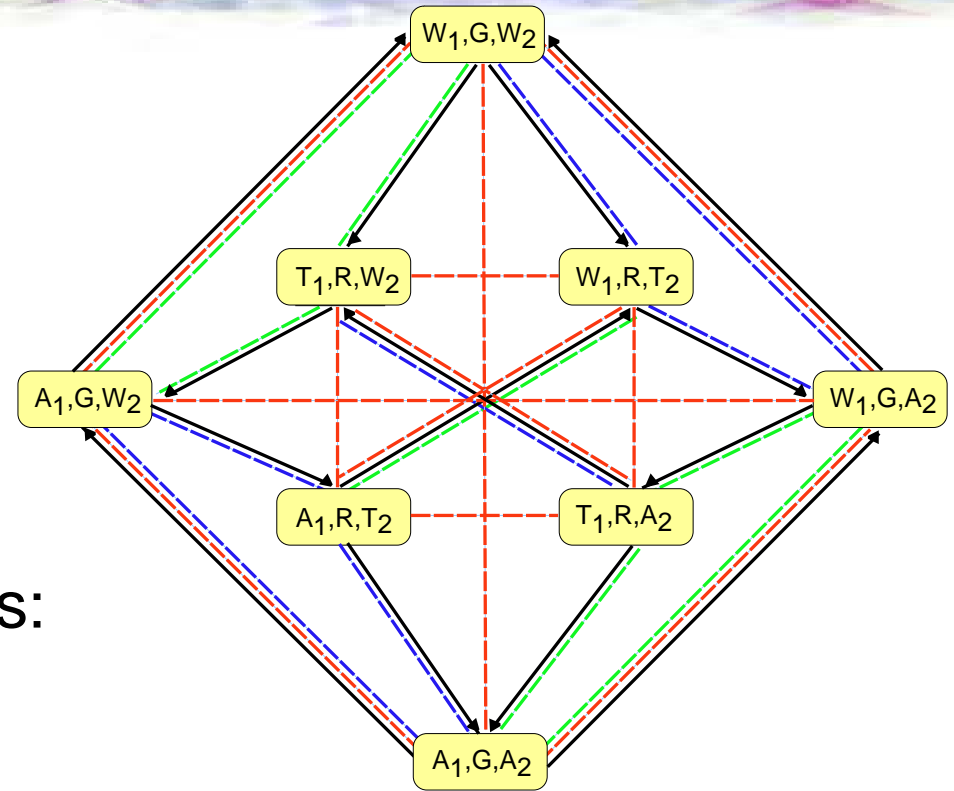


The global state space G
and the successor relation

Interpreted Systems - Example of Trains from [M. Wooldridge]



Local transition structures:
Two trains and a tunnel



- transition relation
- - - epistemic relation for controller
- - - epistemic relation for train₁
- - - epistemic relation for train₂

Reachable global states
and epistemic relations

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

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

$$\varphi = (v_1 \vee v_8 \vee \neg v_2) \wedge (\neg v_1) \wedge (v_4 \vee \neg v_2)$$

Idea of Bounded Model Checking

(**BMC**)

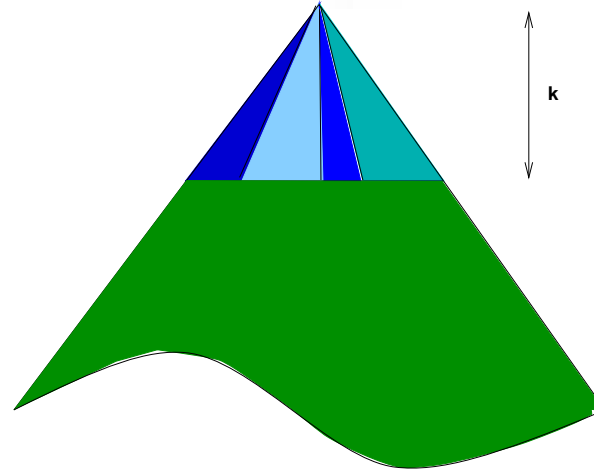
- ✧ **BMC**: to prove that an **ECTLpK** formula holds or that an **ACTLpK** formula does not hold in M_P
- ✧
 - ✧ If $\psi \in \mathbf{ACTLpK}$, take negation $\varphi := \neg\psi$
 - ✧ If $\psi \in \mathbf{ECTLpK}$, $\varphi := \psi$
 2. Take a fragment M' of the model M_P
(preserving φ , i.e., $M' \models \varphi$ implies $M_P \models \varphi$),
 3. Translate $M' \models \varphi$ to a Boolean formula
 $[M'] \wedge [\varphi]_{M'}$, ($M' \models \varphi$ iff $[M'] \wedge [\varphi]_{M'}$ is satisfiable),
 4. Check the satisfiability of $[M'] \wedge [\varphi]_{M'}$.

Conclusion:

If $[M'] \wedge [\varphi]_{M'}$ is satisfiable, then $M_P \models \varphi$.

BMC for an ECTLpK formula φ

- ✦ Let φ be an ECTLpK formula,
- ✦ Iterate for $k := 1$ to $|M|$,
- ✦ Select the k -model M_k ,
- ✦ Select the $f_k(\varphi)$ -submodels of M_k (of $f_k(\varphi)$ paths),
- ✦ Translate the transition relation of the k -computations of M_k to a propositional formula $[M^{\varphi, \iota}]_k$,
- ✦ Translate φ over all the $f_k(\varphi)$ -submodels to a propositional formula $[\varphi]_{M_k}$,
- ✦ Check the satisfiability of $[M, \varphi]_k := [M^{\varphi, \iota}]_k \wedge [\varphi]_{M_k}$.



Selecting submodels of the k -model

- ✦ $M \models \varphi$ iff $[M^{\varphi, \mathcal{L}}]_k \wedge [\varphi]_{M_k}$ is satisfiable.
- ✦ SAT solver: used directly, unchanged
- ✦ Limitations: restricted to falsifying **ACTL_pK** or proving **ECTL_pK** formulas.
- ✦ BMC is incomplete in practice

Idea of Unbounded Model Checking for CTLpK

$M_P \models \varphi$ iff $[\varphi](w) \wedge I_\iota(w)$ is satisfiable,

- w - vector of propositional variables representing a global state,
- $[\varphi](w)$ - symbolic encoding of the set of states in which φ holds,
- $I_\iota(w)$ - symbolic encoding of the initial state ι .

Translation of CTLpK formulas into propositional formulas

Exploit two basic algorithms.

1. The procedure *forall* is applied to formulas $Z\alpha$ s.t. $Z \in \{AX, AY, \mathcal{K}_i, D_r, E_r\}$.
This procedure eliminates the universal quantifier from a QBF formula representing the formula $Z\alpha$ and returns the result in CNF.
2. The procedures *gfp_Z* is applied to formulas $Z\alpha$ s.t. $Z \in \{AG, AH, \mathcal{C}_r\}$.
This procedure uses the fixed point characterisation of the formula $Z\alpha$.

Fixed point characterization

$$\langle AG\alpha \rangle = \nu Z. \langle \alpha \rangle \cap AX(Z),$$

$$\langle AH\alpha \rangle = \nu Z. \langle \alpha \rangle \cap AY(Z),$$

$$\langle C_r\alpha \rangle = \nu Z. E_r(\langle \alpha \rangle \cap Z).$$

Algorithm forall [McMillan 2002]

- ✧ An epistemic formula is encoded in **QBF** as $(\forall a)\varphi(a)$, but we need propositional formulas ...
- ✧ The algorithm *equCNF* translates a Boolean formula φ to an equivalent CNF formula.
 - ✧ $\varphi = a \vee (b \wedge c) \xrightarrow{\text{equCNF}} \varphi_{CNF} = (a \vee b) \wedge (a \vee c)$
Universally quantified variables are removed,
 - ✧ $(\forall a)\varphi(a) \equiv (\forall a)\varphi_{CNF}(a) \equiv (\cancel{a} \vee b) \wedge (\cancel{a} \vee c) = b \wedge c$
- ✧ *equCNF* - realized in a modified **SAT** algorithm: each satisfying assignment (solution of the modified **SAT**) adds a new clause
- ✧ On-the-fly quantifiers elimination

Encoding for UMC

$$[p]_{\mathcal{M}}(w) := \bigvee_{s \in \langle p \rangle} I_s(w), \text{ for } p \in \mathcal{PV}_{\mathcal{K}},$$

$$[\neg\alpha]_{\mathcal{M}}(w) := \neg[\alpha]_{\mathcal{M}}(w),$$

$$[\alpha \wedge \beta]_{\mathcal{M}}(w) := [\alpha]_{\mathcal{M}}(w) \wedge [\beta]_{\mathcal{M}}(w),$$

$$[\alpha \vee \beta]_{\mathcal{M}}(w) := [\alpha]_{\mathcal{M}}(w) \vee [\beta]_{\mathcal{M}}(w),$$

$$[AX\alpha]_{\mathcal{M}}(w) := \text{forall}(\forall v, (T(w, v) \Rightarrow [\alpha]_{\mathcal{M}}(v))),$$

$$[AY\alpha]_{\mathcal{M}}(w) := \text{forall}(\forall v, (T(v, w) \Rightarrow [\alpha]_{\mathcal{M}}(v))),$$

Encoding for UMC - cont'd

- $[\mathcal{K}_i\alpha]_{\mathcal{M}}(w) := \text{forall}(\forall v, ((H_i(w, v) \wedge \neg \text{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathcal{M}}(v))),$
- $[D_\Gamma\alpha]_{\mathcal{M}}(w) := \text{forall}(\forall v, ((\bigwedge_{i \in \Gamma} H_i(w, v) \wedge \neg \text{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathcal{M}}(v))),$
- $[E_\Gamma\alpha]_{\mathcal{M}}(w) := \text{forall}(\forall v, ((\bigvee_{i \in \Gamma} H_i(w, v) \wedge \neg \text{gfp}_{AH}(\neg I_\iota(v))) \Rightarrow [\alpha]_{\mathcal{M}}(v))),$
- $[AH\alpha]_{\mathcal{M}}(w) := \text{gfp}_{AH}([\alpha]_{\mathcal{M}}(w)),$
- $[C_\Gamma\alpha]_{\mathcal{M}}(w) := \text{gfp}_{C_\Gamma}([\alpha]_{\mathcal{M}}(w)).$

Procedure *gfp*

```
procedure gfpCF([ $\alpha$ ](w)), where  $\alpha$  is an CTLpK formula
let  $Q(w) = [true](w)$ 
let  $Z(w) = forall(\forall v, ((\bigvee_{i \in \Gamma} H_i(w, v) \wedge \neg gfp_{AH}(\neg I_i(v))) \Rightarrow [\alpha](v)))$ 
while  $\neg(Q(w) \Rightarrow Z(w))$  is satisfiable
    let  $Q(w) = Z(w)$ ,
    let  $Z(w) = forall(\forall v, (\bigvee_{i \in \Gamma} H_i(w, v) \wedge \neg gfp_{AH}(\neg I_i(v)) \Rightarrow (Z(v) \wedge [\alpha](v))))$ 
return  $Q(w)$ 
```

Trains - a property

$\alpha(N)$: if train 1 is in the tunnel, then it knows whether there is another train in the tunnel.

ACTLpK formula:

$$\alpha(N) = AG(tunnel_1 \Rightarrow (K_{train_1} \bigvee_{i=2..N} tunnel_i))$$

ECTLpK formula:

$$\neg\alpha(N) = EF(tunnel_1 \wedge \bar{K}_{train_1} \bigwedge_{i=2..N} \neg tunnel_i)$$

$$M_P \models \neg\alpha(N)$$

Preliminary experiments with BMC and UMC

Results of BMC:

Number of trains	depth	Size of formula (vars/clauses)	Solver time
10	1	453/1195	0
	2	811/2199	0
50	1	3233/8895	0.01
	2	5971/16759	0.32
100	1	8958/25270	0.03
	2	16921/48459	2.43
300	1	56858/165770	0.32
	2	110721/325259	60.34

Results of UMC - 3 trains: 7 seconds, 5 trains: 300 seconds, 6 trains: 30 mins.

Conclusions and Future Work

- ✦ Two complementary SAT-based methods developed,
- ✦ UMC needs optimizations,
- ✦ We plan to extend VerICS with a high-level language for specifying MAS,
- ✦ Looking for new case studies.

Home page of VerICS:

www.ipipan.waw.pl/~penczek/verics