Strategic and Symbolic Methods for Real-Time Systems in Maude

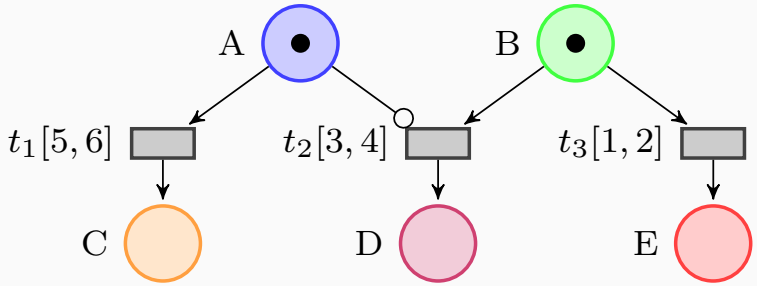
Carlos Olarte

LIPN-Université Sorbonne Paris Nord.

Joint work with J. Arias, K. Bae, P. Ölveczky and L. Petrucci.

Motivation: Verification of real-time systems

- Timed automata
- Time Petri nets



Pros

- Decidable fragments
- Efficient verification procedures
- Extended with parameters.

Cons

1. No support for user-defined data types
2. No support for other forms of communication and dynamic object creation/deletion

Motivation: Verification of real-time systems

Rewriting logic / Maude

```
mod SYSTEM
  eq t = t' .
  rl l => r if C .
  ...
```

Pros

- Very expressive and general
- User-defined data types
- Large applications
- Executable specification
- Maude system: full LTL model checking, reachability, ...

Cons

- Most analysis problems are undecidable
- Explicit-state analysis of real-time theories is unsound for dense time

In this talk

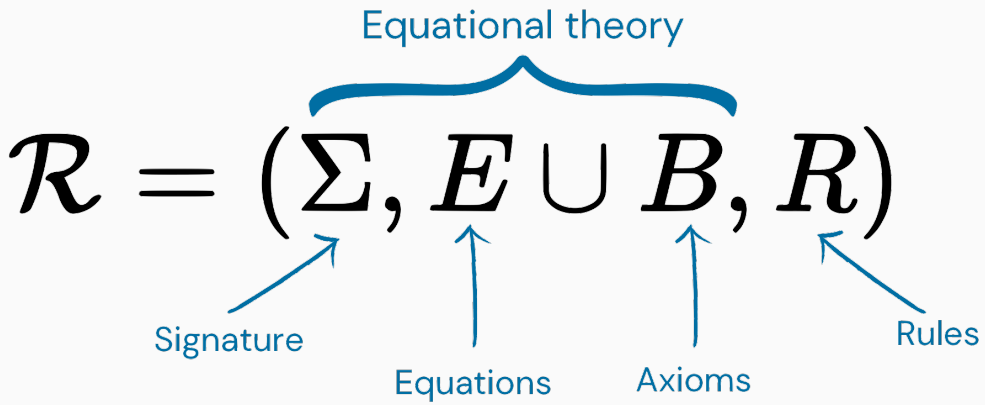
- Interpreter: Executable rewrite semantics for parametric timed Automata (PTA) and parametric time Petri nets with inhibitor arcs (PITPN)
- Sound and complete symbolic analysis (Rewriting + SMT)
- Novel folding procedure (termination)
- Analysis methods: Reachability, parameter synthesis, model checking.
- Strategies I: controlling the execution of real-time theories.
- Strategies II: model checking real-time multi-agent systems (STCTL).

Outline

1. PTAs
2. PITPNs
3. Verification of real-time multi-agent systems
4. Concluding Remarks

Rewriting Logic

- Computational logic: concurrent computation + logical deduction



- States are terms modulo $E \cup B$
- Rules (`crl l => r if C`) in R define system transitions

Rewriting Logic

- Equational theory (algebraic data types): defining states
- Rewriting rules: behavior of the system
- Executable specification

- A high-performance rewriting logic engine
- Executes admissible theories (confluence and termination of *E*, coherence of *R* w.r.t. *E*, ...)
- Several generic formal analysis tools (rewrite, search, LTL model checker, narrowing, SMT, etc).

Plan

1. PTAs
2. PITPNs
3. Verification of real-time multi-agent systems
4. Concluding Remarks

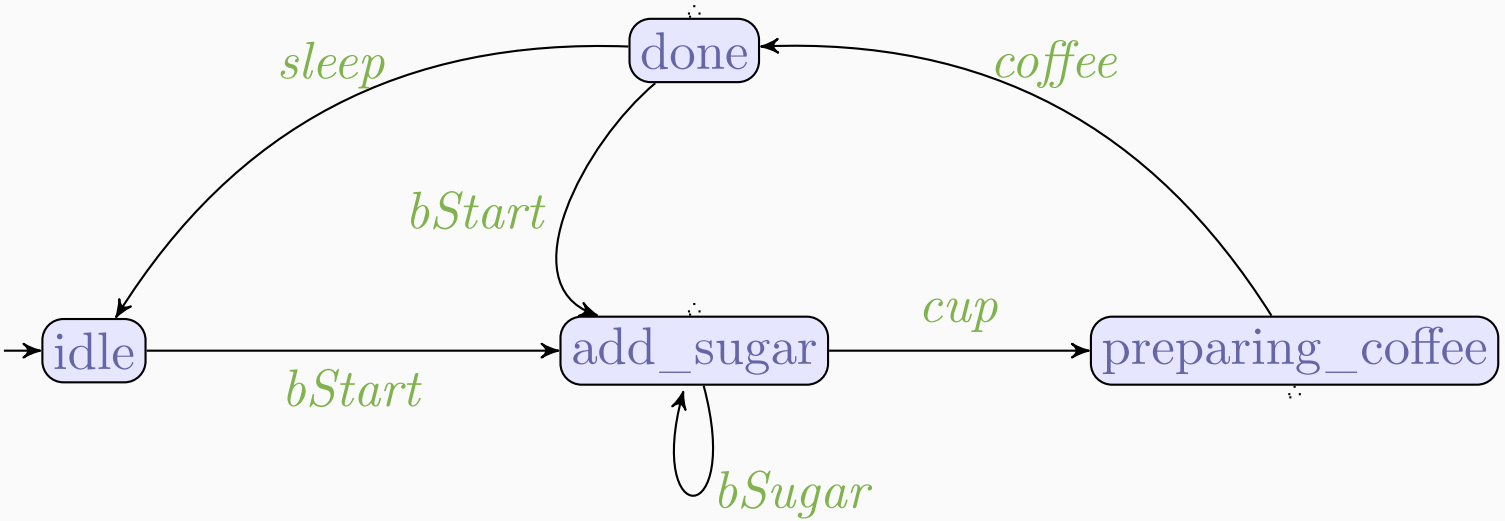
Parametric Verification

Models: not all the choices, components, response times, are known.

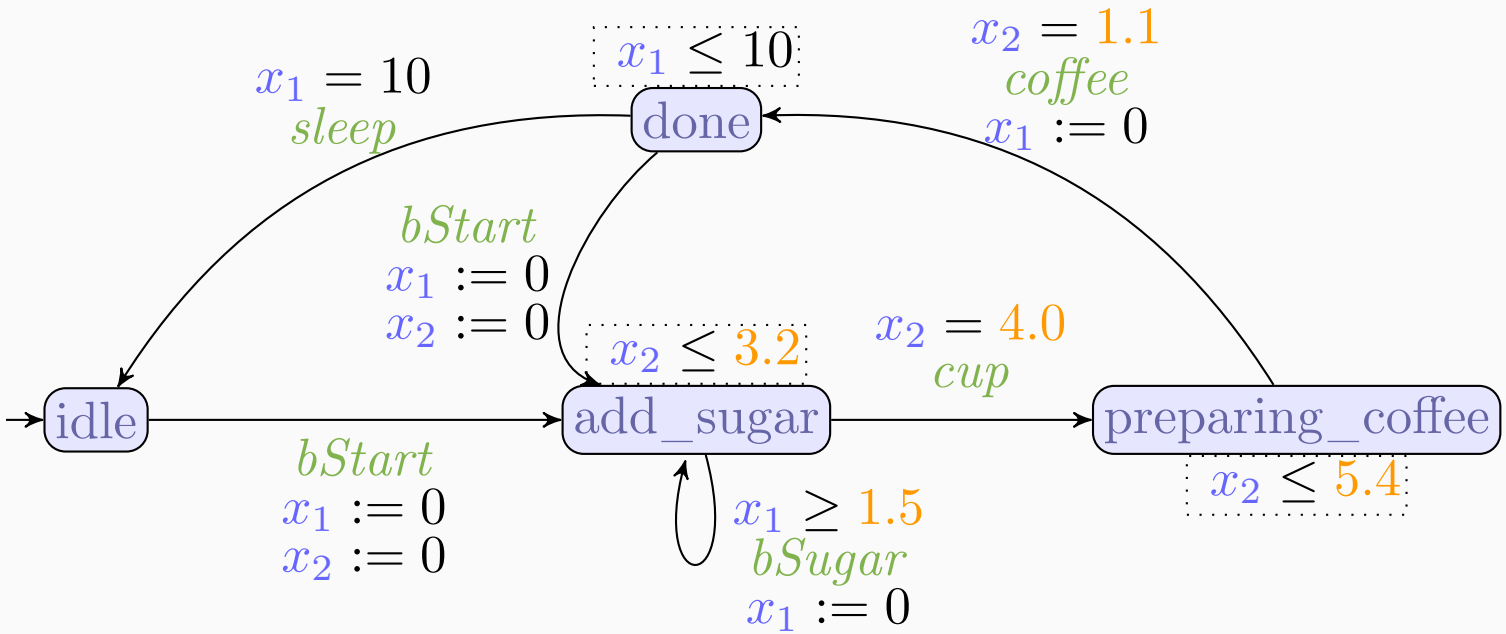
Parameters

- Flexibility
- Avoid verifying the system when the unknown components change
- Central problem: Parameter synthesis

Automata

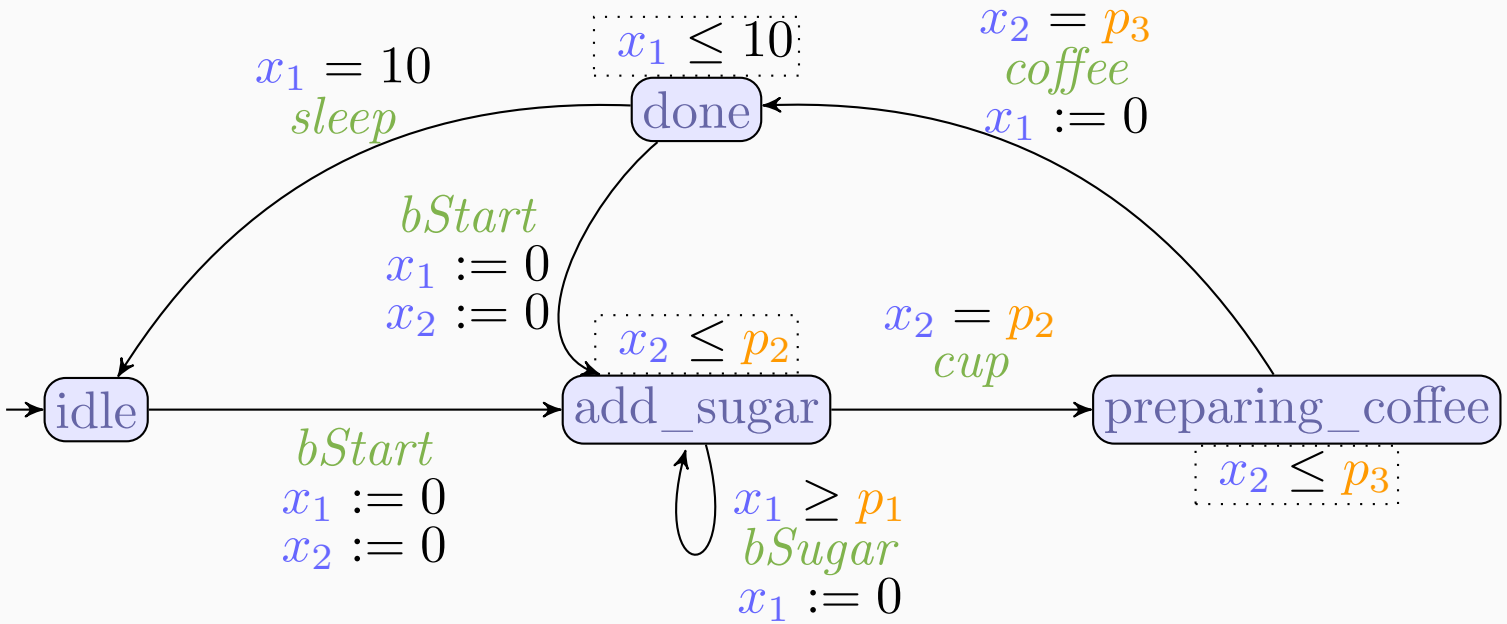

done
sleep
coffee
bStart
idle
add_sugar
cup
preparing_coffee
bStart
bSugar

Timed Automata



clocks and constants
Constraints for invariants and guards.

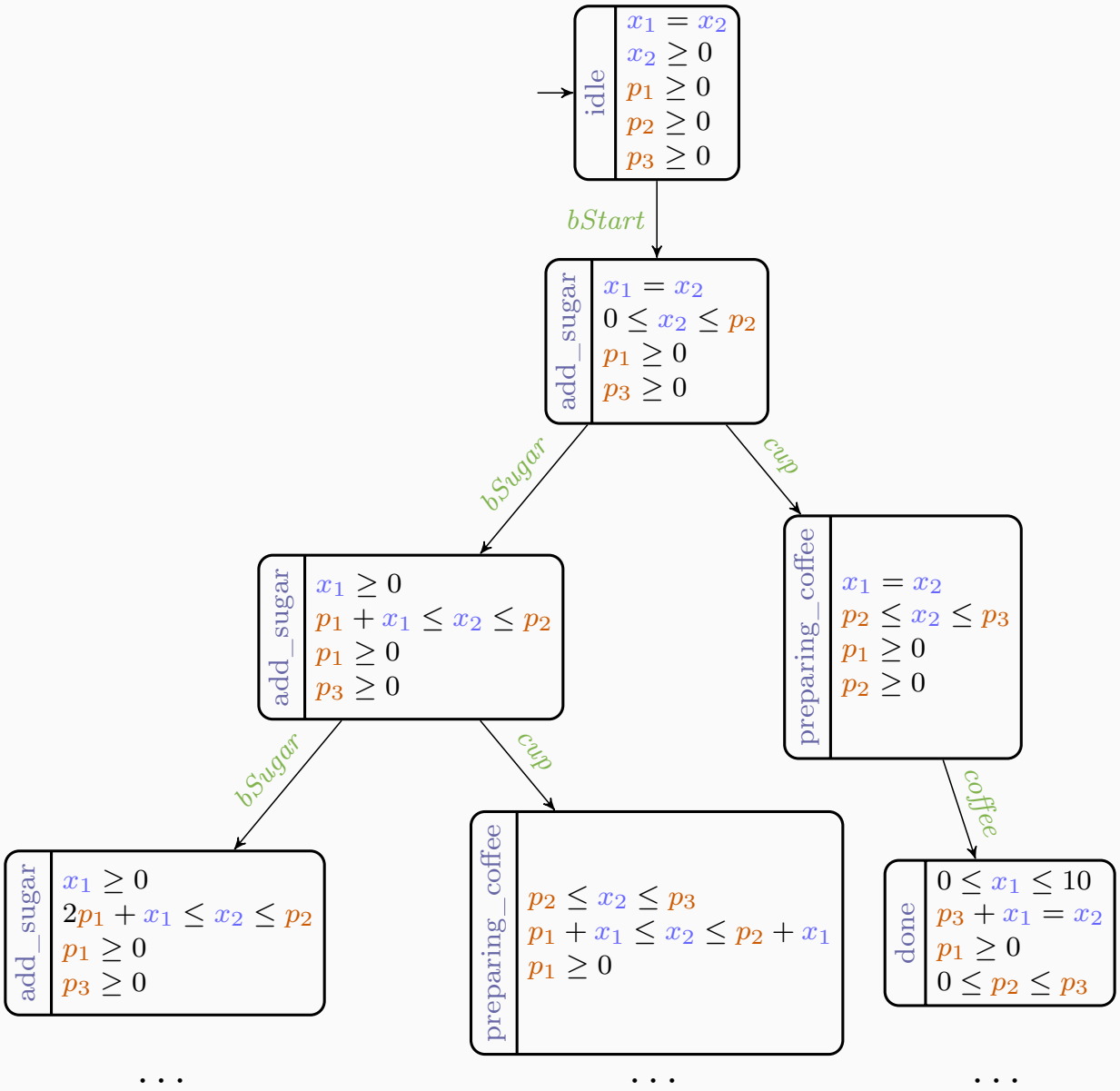
Parametric Timed Automata



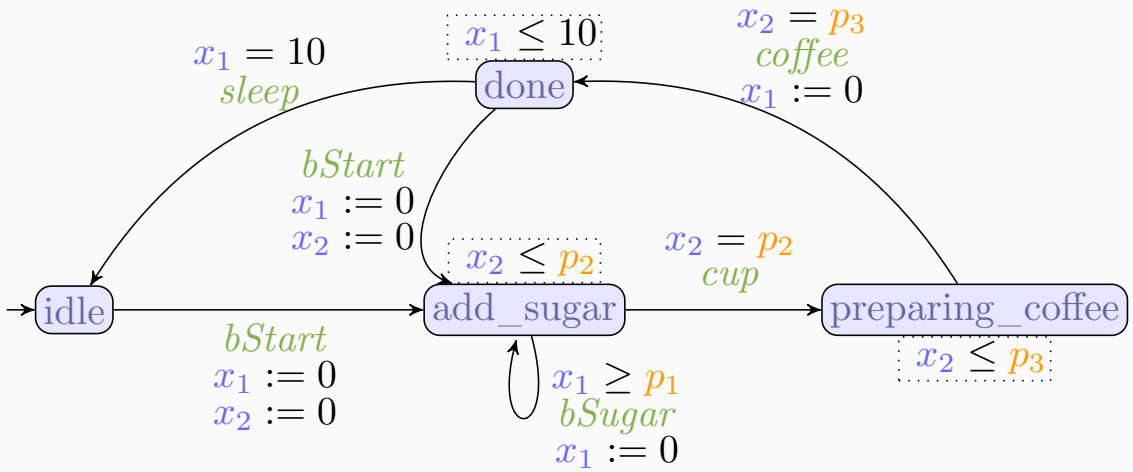
clocks and parameters

Synthesis: If $p_2 \leq p_3$ we can have a coffee.

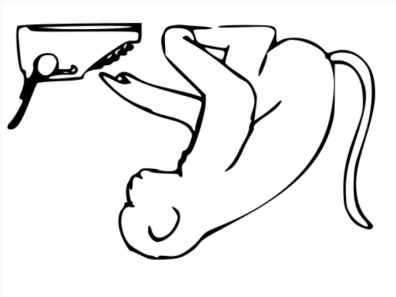
Parametric Zone Graph (PZG)



Parametric Timed Automata



Imitator



- Timed model checking
- Parameter synthesis with dedicated algorithms
- Restricted to PTA-based systems

PTA/Imitator vs Rewriting Logic/Maude

PTA/Imitator

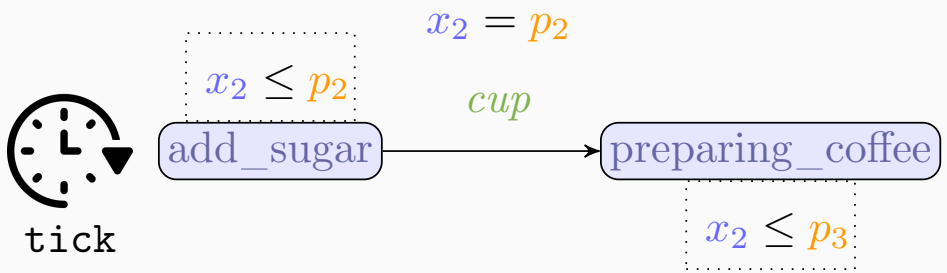
- Heuristic, optimizations and approximation techniques.
- Decidable fragments

RL/Maude

- More expressive (datatypes)
- Undecidable in general
- No parametric analysis

Learn and take the best from both worlds.

RW semantics for PTA: The tick rule problem

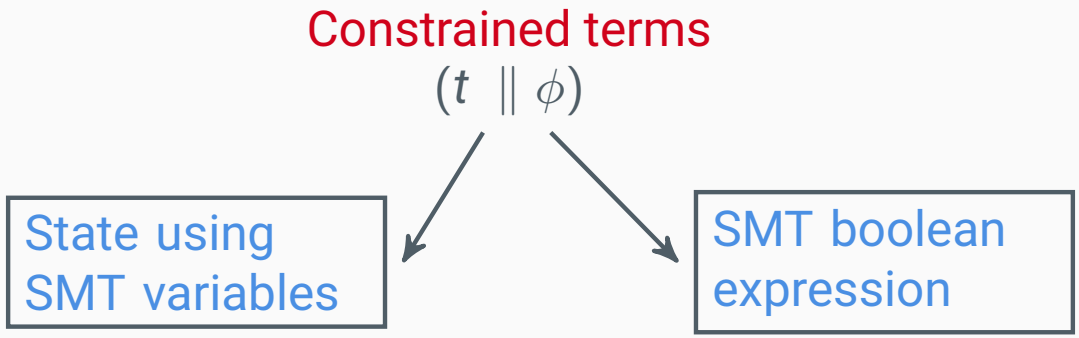



```
vars T P1 P2 P3 : PosRat .
  crl [tick] :
      [ add_sugar : X1 ; X2 ] < P1 ; P2 ; P3 > =>
      < add_sugar : X1 + T ; X2 + T > < P1 ; P2 ; P3 >
      if (X2 + T <= P2 and T >= 0/1) = true [nonexec] .
```

- What is the value of `T`? Non executable rule.
- Sampling is useful but not sound and complete.
- Parameters P_1, P_2, P_3 are indeed constants (ground rewriting)

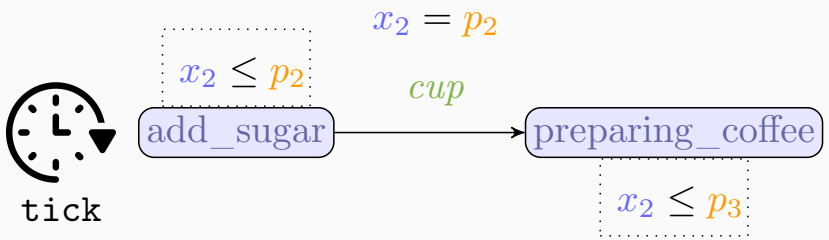
Solution: Rewriting Modulo SMT.

Symbolic semantics: rewriting modulo SMT



- $[\![(t \parallel \phi)]\!]$: possibly infinite set of concrete states (instances of t)
- Symbolic rewrite relation $\rightsquigarrow$: adding constraints + checking satisfiability
- A single symbolic transition captures all possible delays.

RW semantics for PTA



Symbolic states: $t \parallel \phi$ (term + SMT boolean expression)

```
var T P1 P2 P3 : Real .
crl [add_sugar-tick] :
    [ add_sugar : X1 ; X2 ] < P1 ; P2 ; P3 > =>
    < add_sugar : X1 + T ; X2 + T > < P1 ; P2 ; P3 >
    if (X2 + T <= P2 and T >= 0/1) = true [nonexec] .
```

Theorem (Adequacy)

Symbolic executions ($\rightsquigarrow$) correspond to transitions of the PZG

PTA in Maude: Verification techniques

- Reachability analysis:

```
smt-search [1] < idle ; X ; y >   < P1 ; P2 ; P3 >  =>*
               < done ; X' ; X' > < P1 ; P2 ; P3 >
                such that ψ .
```

Parameter synthesis: accumulated constraint $t \parallel \phi$:

- EFϕ synthesis: reachability plus quantifier elimination $\exists X.\phi$ (with Z3).
- AG$\neg\phi$: finding all solutions (termination problem) and negating the result.

Termination problem

- Each `tick` creates a fresh SMT variable
- Maude+SMT analyses do not terminate (even if the PZG is finite)
- The standard subsumption relation is not sufficient.

$$\texttt{State 1}: \quad \langle \texttt{clock}:x \rangle \mid x = t_0 \wedge 0 \leq t_0 \leq 2$$
$$\texttt{State n}: \quad \langle \texttt{clock}:y \rangle \mid \ldots \wedge y = t_1 + t_2 \wedge 0 \leq t_1 + t_2 \leq 2$$

Termination problem

- Each `tick` creates a fresh SMT variable
- Maude+SMT analyses do not terminate (even if the PZG is finite)
- The standard subsumption relation is not sufficient.

$$\begin{array}{ll} \texttt{State 1}: & \langle \texttt{clock:}x \rangle \mid x = t_0 \wedge 0 \leq t_0 \leq 2 \\ \texttt{State n}: & \langle \texttt{clock:}y \rangle \mid \ldots \wedge y = t_1 + t_2 \wedge 0 \leq t_1 + t_2 \leq 2 \end{array}$$

- Match the states: $\theta = [x \mapsto y]$.
- Check the implication ($\phi_2 \Rightarrow \phi_1\theta$), however,

$$\ldots y = t_1 + t_2 \ldots \text{does not imply } y = t_0$$

- Both clock values are constrained to be in the interval $[0, 2]$.

Folding Procedure

- New relation $\preceq$: based on matching + existential quantification

$$\begin{array}{ll} \texttt{State 1}: & \langle \texttt{clock}{:}x \rangle \mid x = t_0 \wedge 0 \leq t_0 \leq 2 \\ \texttt{State n}: & \langle \texttt{clock}{:}y \rangle \mid \ldots \wedge y = t_1 + t_2 \wedge 0 \leq t_1 + t_2 \leq 2 \end{array}$$

Becomes ($U \Downarrow_{now}$)

$$\begin{array}{ll} \texttt{State 1}: & \langle \texttt{clock}{:}z \rangle \mid \exists t_0, x.(z = x \wedge x = t_0 \wedge 0 \leq t_0 \leq 2) \\ \texttt{State n}: & \langle \texttt{clock}{:}z \rangle \mid \exists t_0, t_1, t_2, y.(z = y \wedge \ldots \wedge y = t_1 + t_2 \ldots) \end{array}$$

Folding Procedure

- New relation $\preceq$: based on matching + existential quantification

$$\begin{array}{ll} \texttt{State 1}: & \langle \texttt{clock}{:}x \rangle \mid x = t_0 \wedge 0 \leq t_0 \leq 2 \\ \texttt{State n}: & \langle \texttt{clock}{:}y \rangle \mid \ldots \wedge y = t_1 + t_2 \wedge 0 \leq t_1 + t_2 \leq 2 \end{array}$$

  Becomes ($U \Downarrow_{now}$)

$$\begin{array}{ll} \texttt{State 1}: & \langle \texttt{clock}{:}z \rangle \mid \exists t_0, x.(z = x \wedge x = t_0 \wedge 0 \leq t_0 \leq 2) \\ \texttt{State n}: & \langle \texttt{clock}{:}z \rangle \mid \exists t_0, t_1, t_2, y.(z = y \wedge \ldots \wedge y = t_1 + t_2 \ldots) \end{array}$$

- We hide the "irrelevant" information (tick variables).
- Let $U \Downarrow_{now} = t_u \parallel \phi_u$ and $V \Downarrow_{now} = t_v \parallel \phi_v$

$$U \preceq V \text{ iff } t_u = t_v\theta \text{ and } \exists(U \Downarrow_{now}) \Rightarrow \exists(V \Downarrow_{now})\theta$$

- Soundness and completeness (for PTAs): $[\![U]\!] \subseteq [\![V]\!]$ iff $U \preceq V$

Theorem

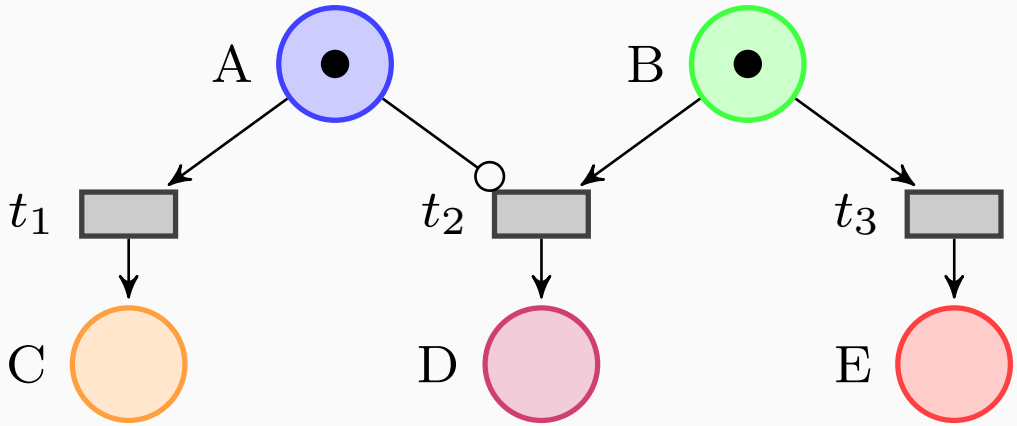
Search with folding terminates iff the PZG is finite.

What we have so far...

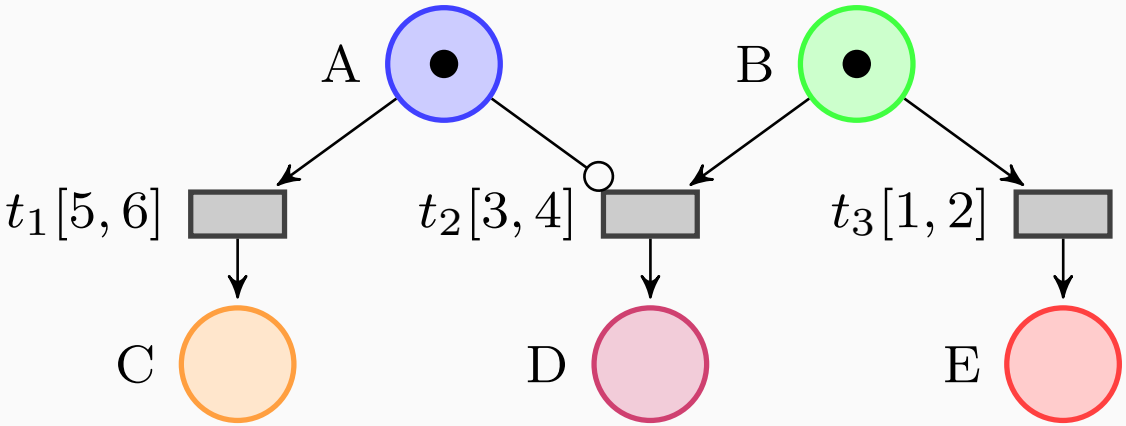
- Relatively simple model (“simple” states)
- Each PTA is compiled into a theory $\mathcal{R}$
- No equations in the theory (if we were to use smt-search)
- With folding: sound and complete reachability analysis
- Parameter synthesis (and folding) only available by calling Z3.

Plan

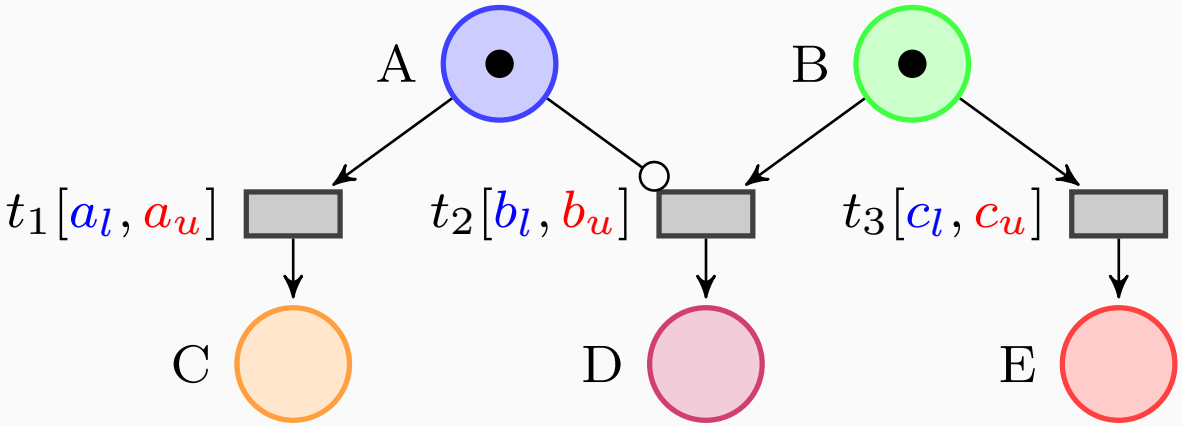
Petri net with inhibitor arcs



Time Petri net with inhinitor arcs



PITPN: Parametric time Petri net with inhibitor arcs



Synthesis problem: finding values for the parameters s.t. certain property holds

Representing PITPNs in Maude

The tick rule: delay transition $\xrightarrow{\delta}$

```
crl [tick] : M : CLOCKS : NET  =>  M : increaseClocks(M, CLOCKS, NET, T) : NET
if  (T >= 0 and mte(M, CLOCKS, NET, T )) = true  [nonexec] .
```

- Enabled transitions are not missed (predicate *mte*)
- *mte* defined equationally.
- "Standard" smt-search cannot be used.
- $\rightsquigarrow$: adding constraints + checking satisfiability

Theorem (Adequacy)

For any PIPTPN $\mathcal{N}$, the symbolic semantics of $\mathcal{N}$ corresponds to $\rightsquigarrow$-transitions.

Theorem (Termination)

Search with folding terminates iff the PZG is finite.

Analyses

EF-synthesis (**EF** ϕ). Standard Maude's search

```
search [1] init(net, m0, φ) =>* S : PHI' || ( TICK : M : CLOCKS : NET )
       such that smtCheck(PHI' and not k-safe(1,M)) .
```

Safety synthesis (**AG**($\neg\phi$)). Search + Folding

```
safety-syn(net, m0, a:Real >= 30/1 and a:Real <= 70/1, k-safe(1,M)) .
```

Strategies. Rewriting + Strategy

New analysis: What happens if t_3 has a higher priority?

```
t3-first := ( applyTransition[L <- "t3"] or-else all )!
```

Analyses

Model Checking. (Search + Maude's LTL model checker)

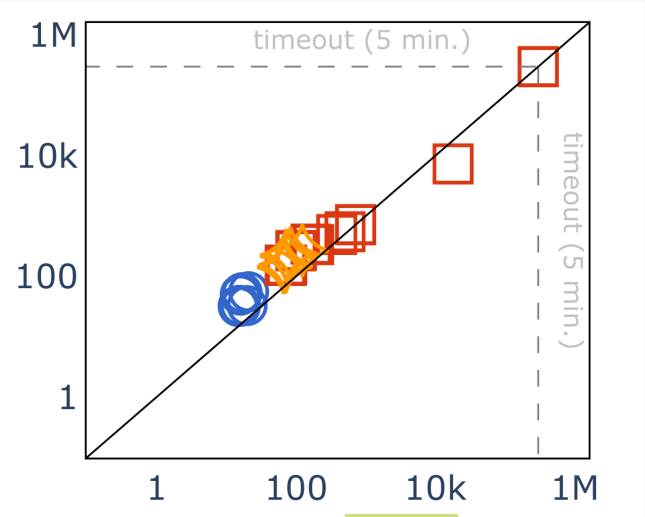
- The TCTL fragment $\exists \mathsf{F}_J\, \phi \mid \forall \mathsf{G}_J\, \phi \mid \phi \rightsquigarrow_{\leq b} \psi$ can be checked with search + folding
- The TCTL fragment $\mathbf{Q}\, \phi\, \mathsf{U}_J\, \psi \mid \forall \mathsf{F}_J\, \phi \mid \exists \mathsf{G}_J\, \phi$ can be checked using Maude's LTL model checker (+ some theory transformations)

Parametric initial marking

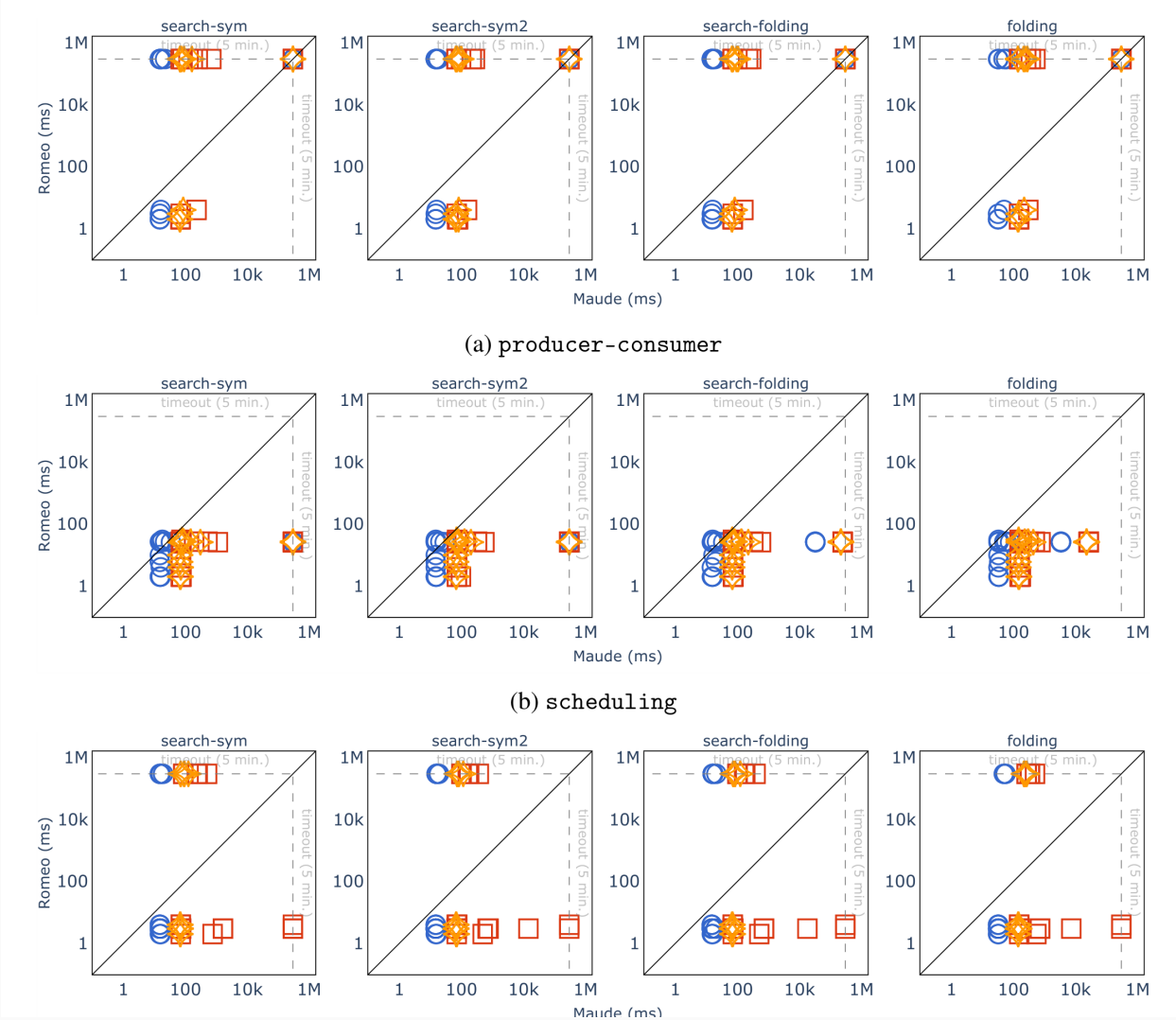
- The number of tokens at a place *p* is an SMT integer variable
- New analysis: Initial marking synthesis

Folding and synthesis

- Quantifier elimination is needed for folding and synthesis.
- Z3 was the only option available for eliminating variables
- We have implemented the FME procedure in Maude
- Yices2 (the fastest in our benchmarks) can be used now.

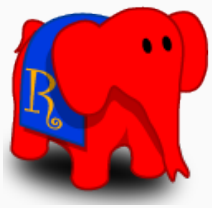


Benchmarks



(a) `producer-consumer`

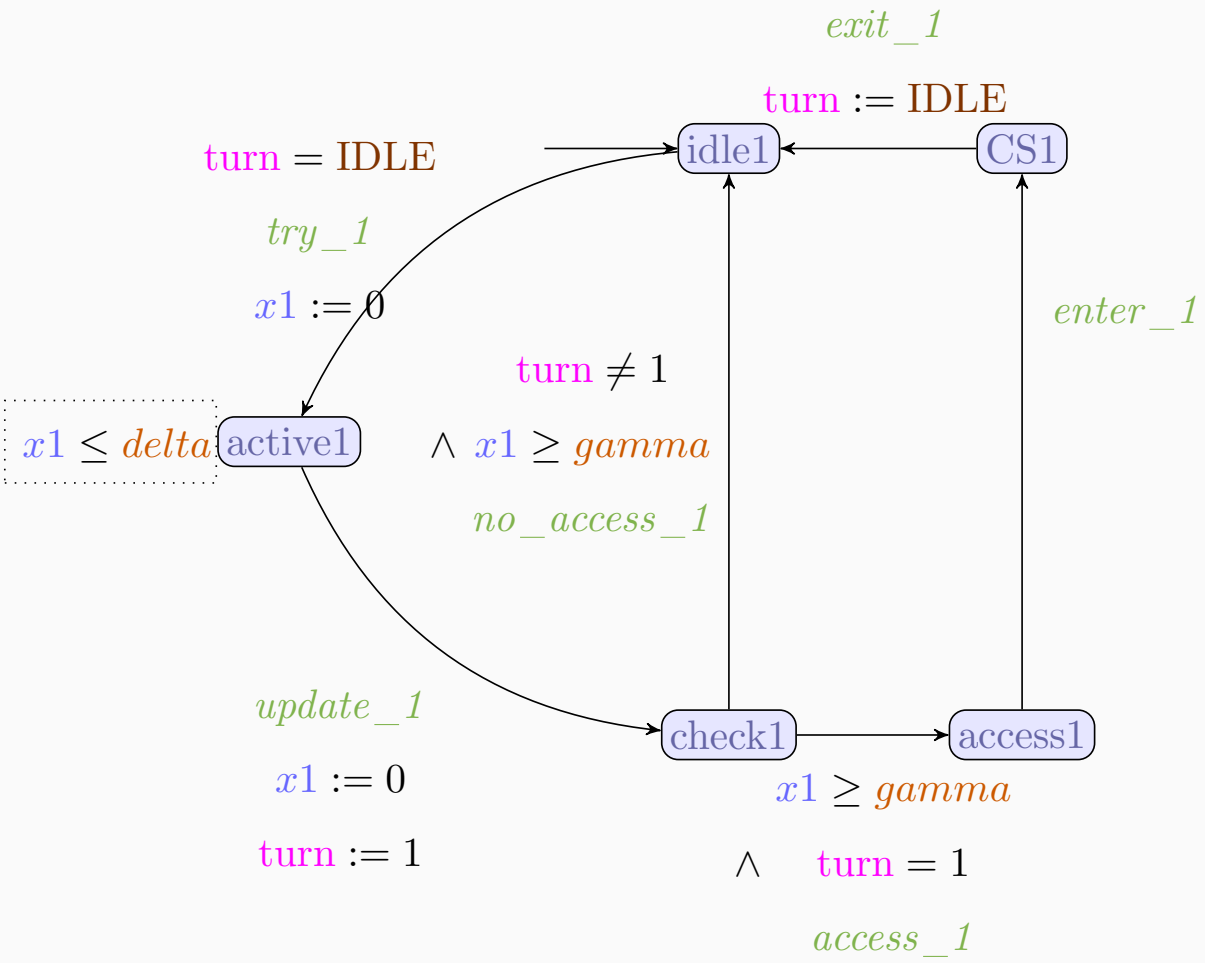
(b) `scheduling`



What we have so far...

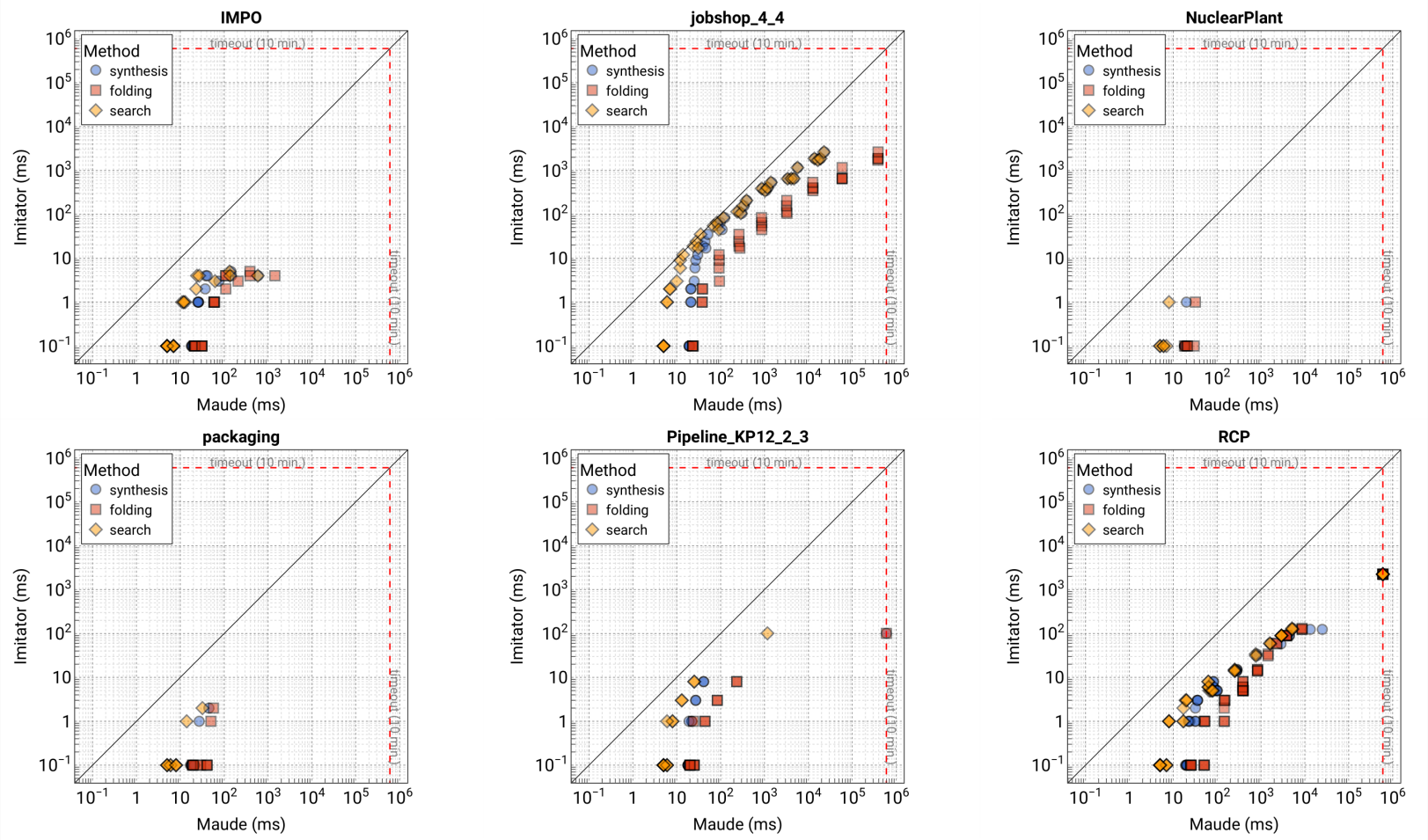
- Equations are part of the theory and `smt-search` cannot be used directly.
- The new folding eliminating tick variables is needed.
- Our benchmarks look much better with our own FME procedure (Yices2).
- Back to PTAs: with the function *mte*$(\cdot)$, we can specify networks of PTAs.

Networks of PTAs with Variables



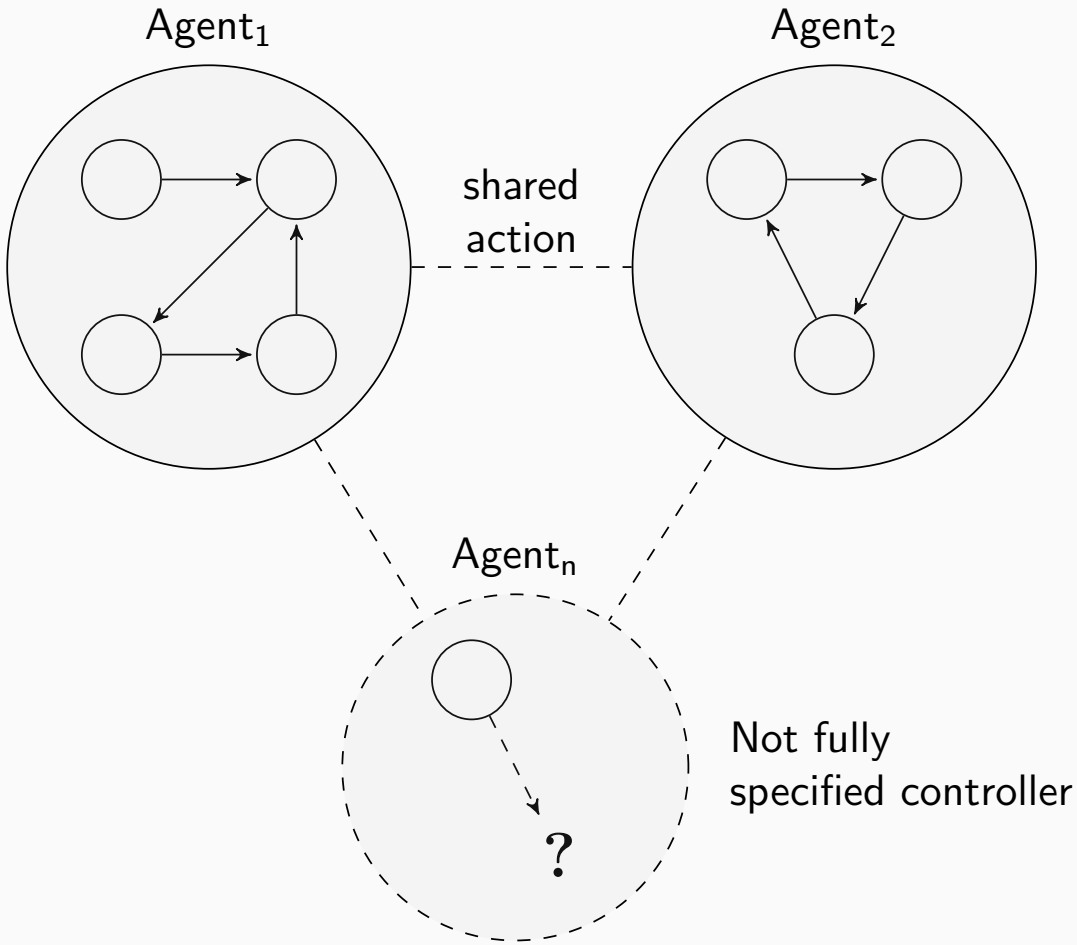
- We need an interpreter with three rules: `tick`, `local`, and `binary`.
- The tick rule checks the invariants in all current locations (*mte*).

Imitator vs Maude



Plan

Networks of parametric timed automata (PTA)

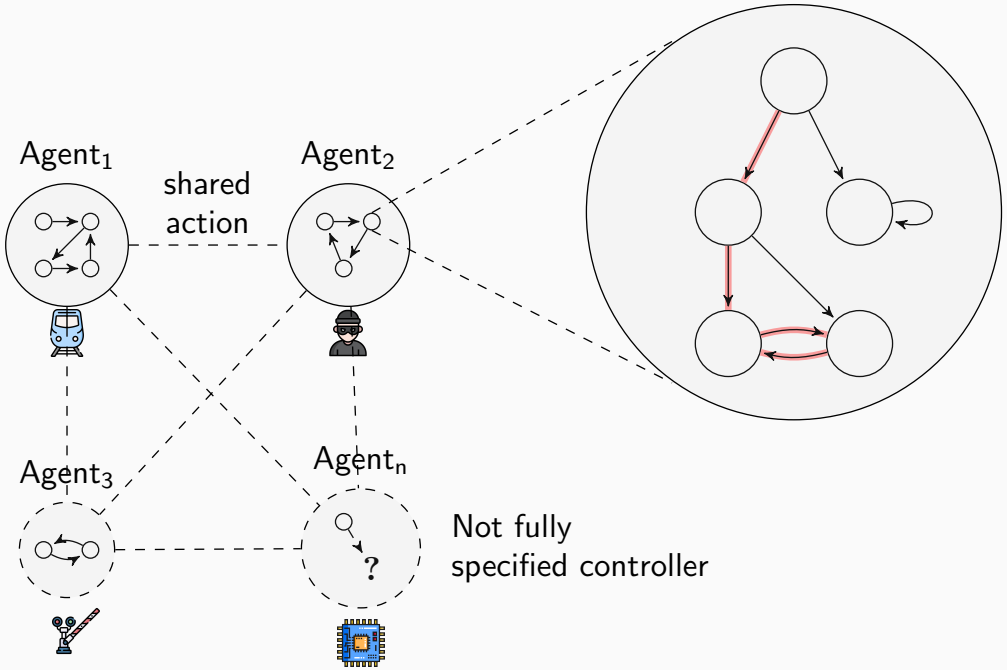


Specification Strategic Timed CTL (STCTL)

- CTL: Existential and universal quantifiers on paths
- TCTL: CTL + Continuous time

Specification Strategic Timed CTL (STCTL)

- CTL: Existential and universal quantifiers on paths
- TCTL: CTL + Continuous time
- STCTL: TCTL + Strategic abilities of agents



- Do the voters have a strategy to elect X?
- Can the attacker obtain the key in less than Y seconds?
- Can the intruder sabotage the system and make the train crash?

Previous approaches

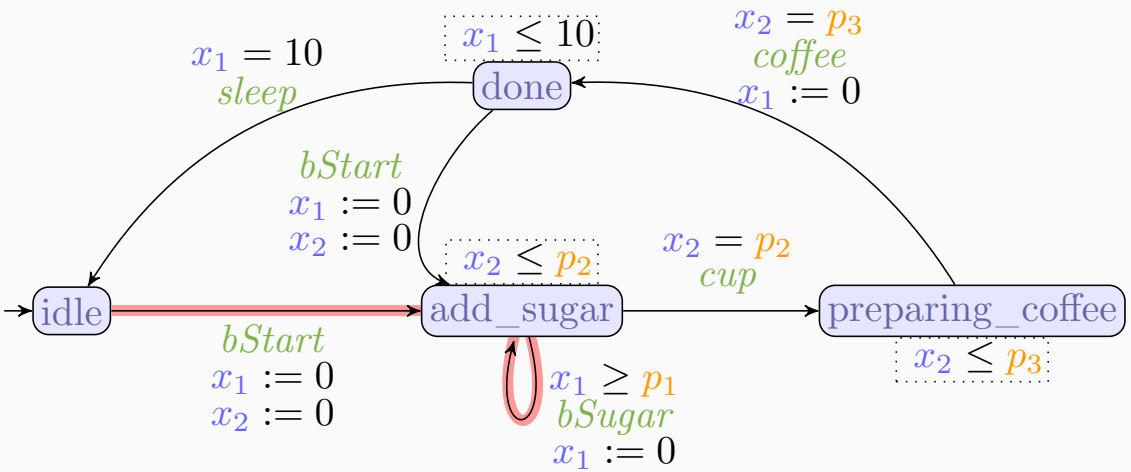
- Limited to the existential and non-nested fragment
- Bounded analysis
- Ad hoc implementations (difficult to reason about correctness)

Strategies in AMAS

- Memoryless imperfect information: $\sigma_i : L_i \rightarrow Act_i$

Strategies in AMAS

- Memoryless imperfect information: $\sigma_i : L_i \to Act_i$



Partial strategy to avoid coffee:

$$\texttt{coffee} : \text{idle do bStart} \to \text{addSugar},$$
$$\texttt{coffee} : \text{addSugar do bSugar} \to \text{addSugar}$$

STCTL

STCTL : CTL + Continuous time + Strategies

$$F, G ::= \mathrm{p} \mid \neg F \mid F \wedge G \mid$$

STCTL

STCTL : CTL + Continuous time + Strategies

$$F, G ::= \mathrm{p} \mid \neg F \mid F \wedge G \mid \langle\langle A \rangle\rangle F$$

- $\langle\langle A \rangle\rangle F$: There is a strategy for coalition A s.t. F

STCTL

STCTL : CTL + Continuous time + Strategies

$$F, G ::= \mathrm{p} \mid \neg F \mid F \wedge G \mid \langle\langle A \rangle\rangle F \mid \mathbf{Q}\mathtt{X}\, F \mid \mathbf{Q} F\, \mathtt{U}_I G \mid \mathbf{Q} F\, \mathtt{R}_I G$$
$$\mathbf{Q} \in \{\exists, \forall\}$$

- $\langle\langle A \rangle\rangle F$: There is a strategy for coalition A s.t. F
- $\forall(F\, \mathtt{U}_{[a,b]} G)$: in all path, G eventually holds in the interval $[a, b]$ and, before that, F remains true.

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$\mathit{VStates} : \mathit{Net} : \mathit{State} : \mathit{Strategy} \models F \text{ i.c. } A$$

VStates
Already visited states (loops and termination)

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$\text{VStates} : \text{Net} : \text{State} : \text{Strategy} \models \text{F i.c. A}$$

Net
Term representing the AMAS

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$\text{VStates} : \text{Net} : \text{State} : \text{Strategy} \models \text{F i.c. A}$$

State
Current state ($\langle \ell_1, \cdots, \ell_n \rangle \parallel \phi$)

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$\mathsf{VStates} : \mathsf{Net} : \mathsf{State} : \mathsf{Strategy} \models \mathsf{F}\ \mathsf{i.c.}\ \mathsf{A}$$

Strategy
Partial strategy for agents in coalition *A*

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$\text{VStates} : \text{Net} : \text{State} : \text{Strategy} \models \text{F i.c. A}$$

Formula
Current formula being checked

Rewriting Semantics for STCTL

We rewrite terms of the form:

$$VStates : Net : State : Strategy \models F \text{ i.c. } A$$

Set of Agents
Current coalition

The Rewriting Strategy solver

```
sd solver := --- By case analysis

  match VS : NET : STATE : STR |= « A' » F i.c. A ?
    (str ; solver)

--- Rule str
rl [str] :  VS : NET : STATE : STR |= « A' » F  i.c. A =>
            VS : NET : STATE : STR |=        F  i.c. A'
```

The Rewriting Strategy solver

```
match VS : NET : STATE : STR |= F ∧ G i.c. A ?
      (and{solver} ; solver)
```

```
--- Rule and
crl [and] :
 VS : NET : STATE              : STR  |= F ∧ G i.c. A =>
 VS : NET : addC(STATE, CONS)  : STR' |=     G i.c. A
if
 VS : NET : STATE : STR |= F i.c. A => Yes(STR',CONS) .
```

The Rewriting Strategy solver

```
match VS : NET : STATE : STR |= EX F i.c. A ?
  move ; EX ; solver
```

The Rewriting Strategy solver

```
  match VS : NET : STATE : STR |= EX F i.c. A ?
    move ; EX ; solver


--- Strategy move
sd move := extend ! ; (local | binary ; --- Next state
                       tick ;
                       subsume) or-else deadlock

--- Rule deadlock
rl [deadlock] : VS : NET : STATE : STR |= F i.c. A =>
                Deadlock(STR, STATE) .
```

The Rewriting Strategy solver

```
  match VS : NET : STATE : STR |= EX F i.c. A ?
    move ; EX ; solver


--- Strategy move
sd move := extend ! ; (local | binary ; --- Next state
                       tick ;
                       subsume) or-else deadlock

--- Rule deadlock
rl [deadlock] : VS : NET : STATE : STR |= F i.c. A =>
                Deadlock(STR, STATE) .

--- The rule EX
rl [EX] : VS : NET : STATE : STR |= EX F i.c. A =>
          VS : NET : STATE : STR |=    F i.c. A .
```

The Rewriting Strategy solver

```
  match VS : NET : STATE : STR |= AX F i.c. A ?
    (neg ; complete ! ; not(solver) ; yes)

--- Rule neg
rl [neg] : VS : NET : STATE : STR |= F   i.c. A =>
           VS : NET : STATE : STR |= !  F i.c. A .

--- Rule yes
rl [yes] : VS : NET : STATE : STR |= F i.c. A =>
           Yes(STR, STATE) .
```

Adequacy

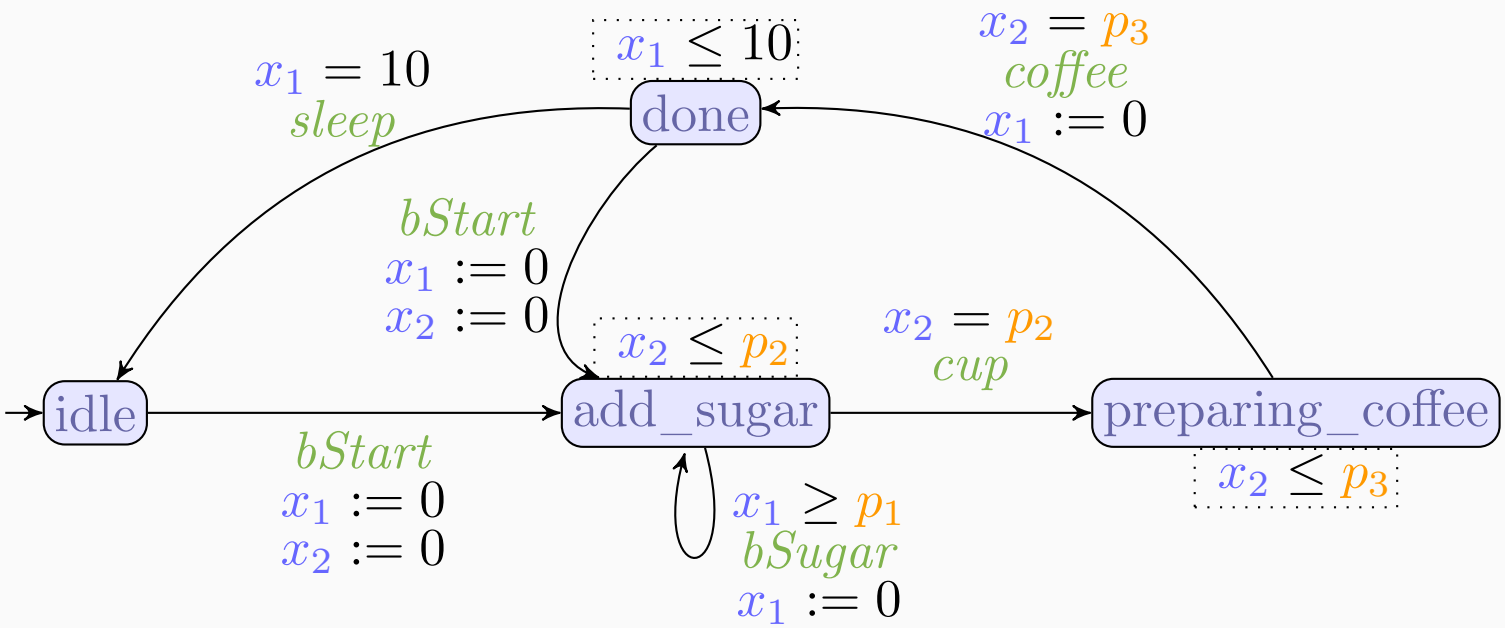
Representational distance: Networks, strategies, formulas, etc. are very close to their corresponding terms of sort `Network`, `Strategy`, `Formula`, etc.

Theorem (Correcteness)

For any AMAS M, initial state ι and formula F:

$$M, \iota \models F \quad \textit{iff}$$
$$\textit{empty}:[\![M]\!]:[\![\iota]\!]:\textit{empty} \models [\![F]\!]\, i\,.\, c\ \textit{empty} \longrightarrow_{\mathcal{R}}^{\textit{solver}} \textit{Yes}(\textit{str}, \phi)$$

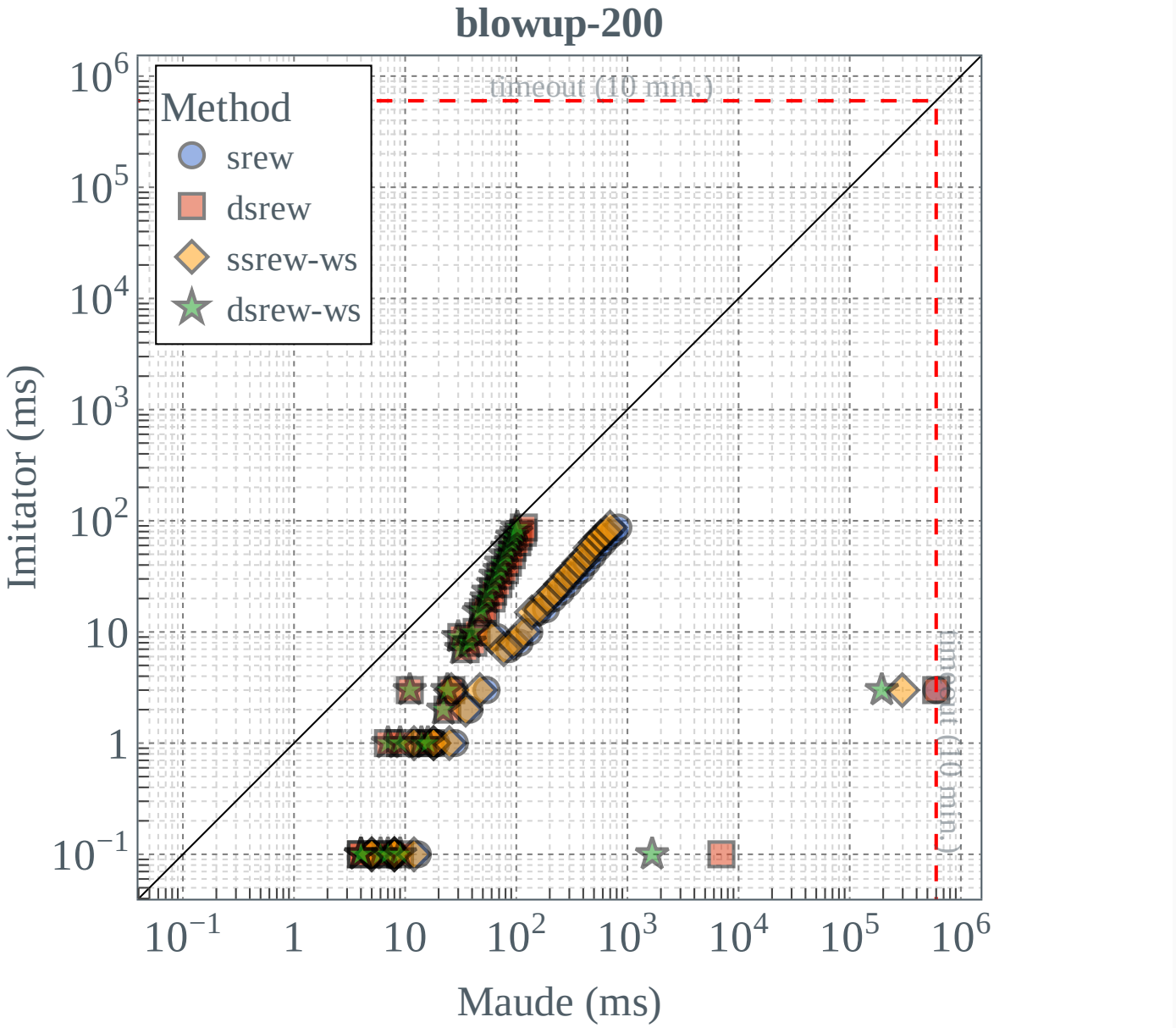
EF-Synthesis and CTL formulas



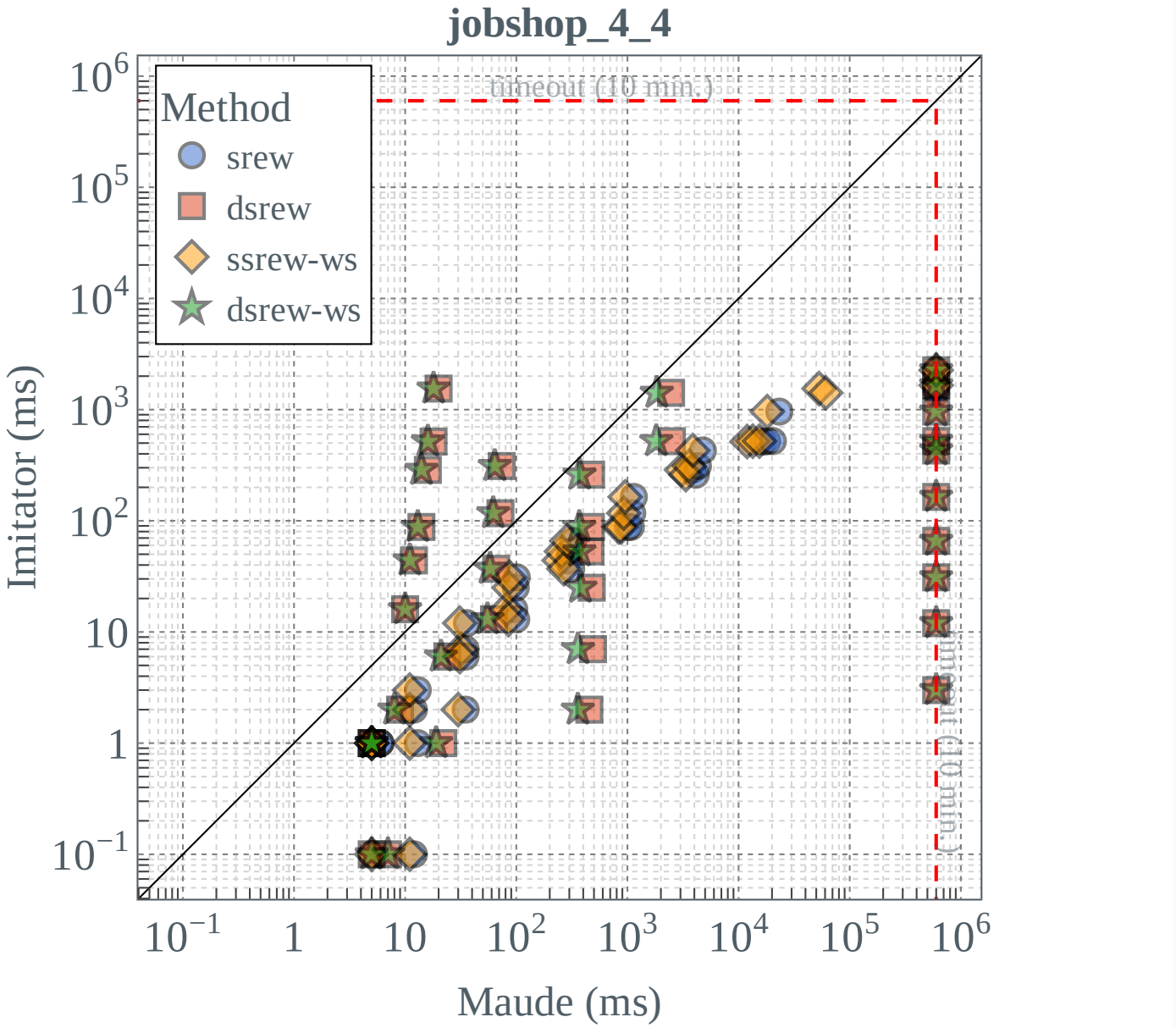

```
Maude> srew wrap(init, << coffee >> (EF (coffee @ done))) using check .

result State: Yes((
    coffee : bstart : idle -> addsugar,
    coffee :  cup :  addsugar -> preparing,
    coffee : coffee : preparing -> done),
    rr(var(p2)) + (-1).NzInt * rr(var( p3)) <= 0 and ...)
```

EF-Synthesis and CTL formulas (Imitator)

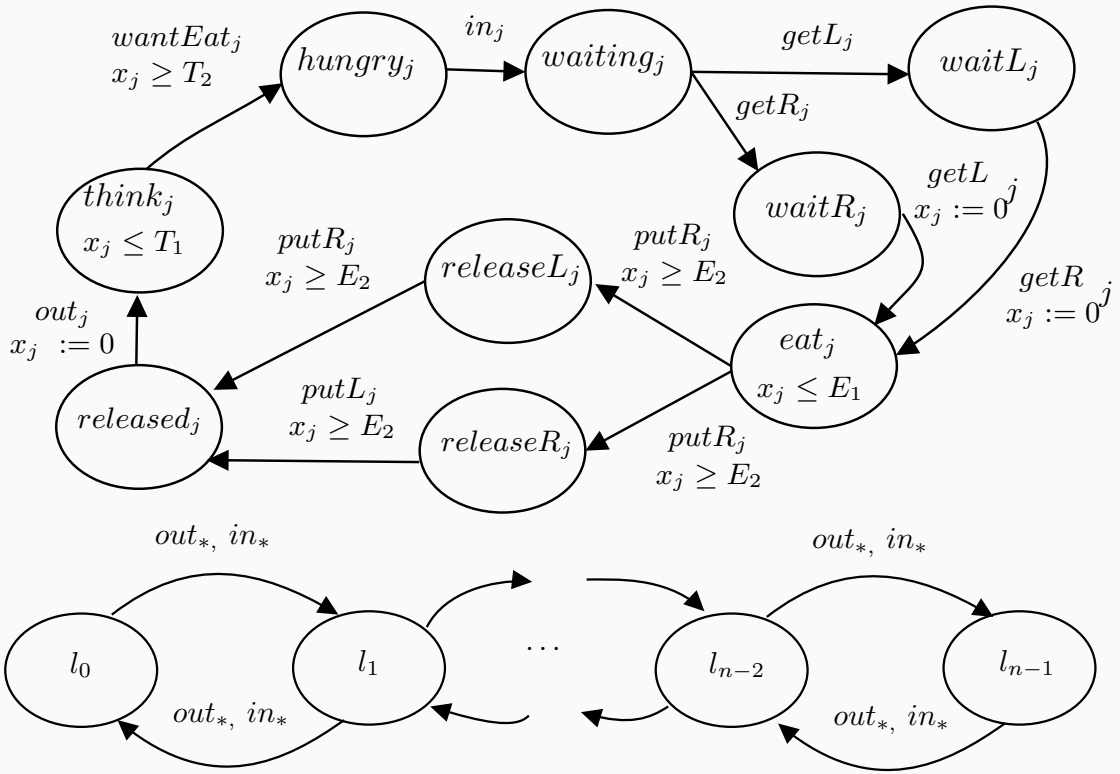


EF-Synthesis and CTL formulas (Imitator)



Maude vs Bounded Model Checking

Timed dining philosopher problem



$$\langle\langle Lackey \rangle\rangle \exists \mathbf{F}_{[1,E_2)} \left(\bigwedge_{j \in odd_p} Eat_j \right)$$

Maude vs Bounded Model Checking

Timed dining philosopher problem

N	Maude(s)	BMC(s)
2	0.02	2
3	0.03	9
4	0.23	118
5	0.28	296
6	13.5	1684
7	333	4081

Maude vs Bounded Model Checking

Timed dining philosopher problem

N	Maude(s)	BMC(s)
2	0.02	2
3	0.03	9
4	0.23	118
5	0.28	296
6	13.5	1684
7	333	4081

Universal Fragment and Synthesis

A more "realistic" specification:

$$\begin{array}{lcl} \mathit{spec} & := & \langle\!\langle \mathit{lackey} \rangle\!\rangle (\mathit{eat} \wedge \mathit{critical}) \\ \mathit{critical} & := & \forall \mathtt{G}\, (\neg \mathit{eat}(0) \vee \neg \mathit{eat}(1)) \\ \mathit{eat} & := & \forall \mathtt{G}\, ((\exists \mathtt{F}\, \mathit{eat}(0)) \wedge (\exists \mathtt{F}\, \mathit{eat}(1))) \end{array}$$

Maude vs Bounded Model Checking

Voting system: $\phi = \langle\langle A \rangle\rangle \exists \mathbf{F}_{[0,8]} v_1$

Instance	$\lvert A \rvert = 1$			$\lvert A \rvert = 2$			$\lvert A \rvert = 3$		
	Imitator	dsrew	srew	Imitator	dsrew	srew	Imitator	dsrew	srew
(13,1)	0.141	0.009	2071	26.423	0.023	**TO**	**TO**	0.042	**TO**
(13,2)	0.223	0.009	2093	41.462	0.023	**TO**	**TO**	0.043	**TO**
(13,3)	0.312	0.009	2272	62.838	0.025	**TO**	**TO**	0.047	**TO**
(13,4)	0.386	0.01	2500	**TO**	0.027	**TO**	**TO**	0.05	**TO**
(15,1)	0.192	0.009	2352	30.682	0.024	**TO**	**TO**	0.044	**TO**
(15,2)	0.304	0.009	2315	57.895	0.024	**TO**	**TO**	0.046	**TO**
(15,3)	0.402	0.01	2494	102.039	0.027	**TO**	**TO**	0.05	**TO**
(15,4)	0.554	0.01	2873	111.891	0.029	**TO**	**TO**	0.053	**TO**

Plan

Concluding Remarks

- Executable symbolic rewrite semantics for PTAs and PITPNs
- Sound and complete analyses: synthesis, reachability, TCTL model checking
- New Analyses: strategies and synthesis of marking.
- Concrete steps for symbolic analysis of real-time rewrite theories
- New strategy language for real-time theories (WRLA'24).

What’s next?

- Full LTL model checking (non-universal properties).
- Integration of SMT and Narrowing.
- All Real-time Maude analyses but symbolically.
- Folding and better integration at the C++ level.
- Other forms of strategies (and other strategic logics).

Thanks!

We acknowledge support from the NATO Science for Peace and Security Programme SymSafe.