# Symbolic Analysis and Parameter Synthesis for Parametric Timed Automata and Petri Nets using Maude with SMT Solving

Peter Ölveczky (University of Oslo)

Based on joint work with Carlos Olarte, Jaime Arias, Laure Petrucci (Sorbonne Paris-Nord); Kyungmin Bae (POSTECH); and Fredrik Rømming (Cambridge)

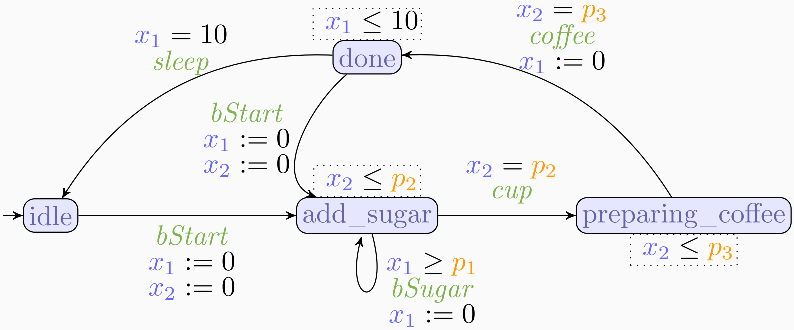
IFIP WG 1.3 Meeting, Salzburg, Feb 5–8, 2024

## Content

# Background

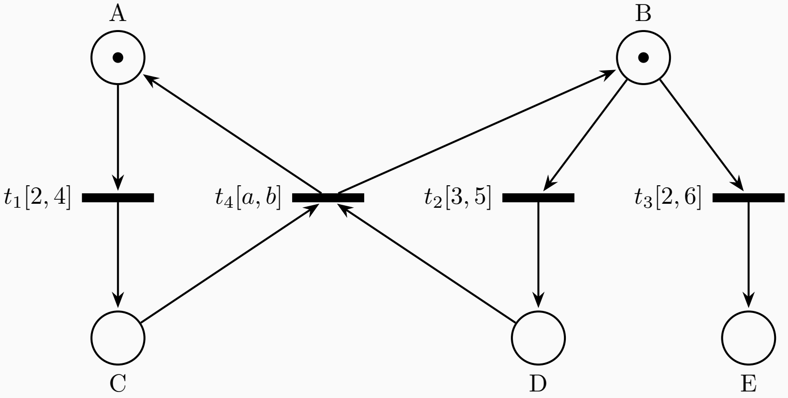
# Context: Formal Models for Real-Time Systems

- Timed automata



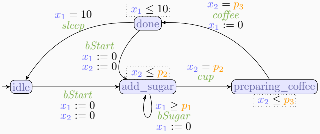
# Context: Formal Models for Real-Time Systems

- Timed automata
- Time(d) Petri nets

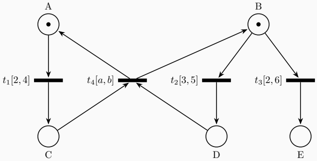


# Context: Formal Models for Real-Time Systems

- Timed automata




- Time(d) Petri nets




- Limited expressiveness/modeling convenience
    - data types; unbounded data structures; fixed communication; ...

\+ many properties decidable

## Context: Real-Time Maude (Ölveczky and Meseguer)

- Expressiveness and modeling convenience
  - algebraic specification!

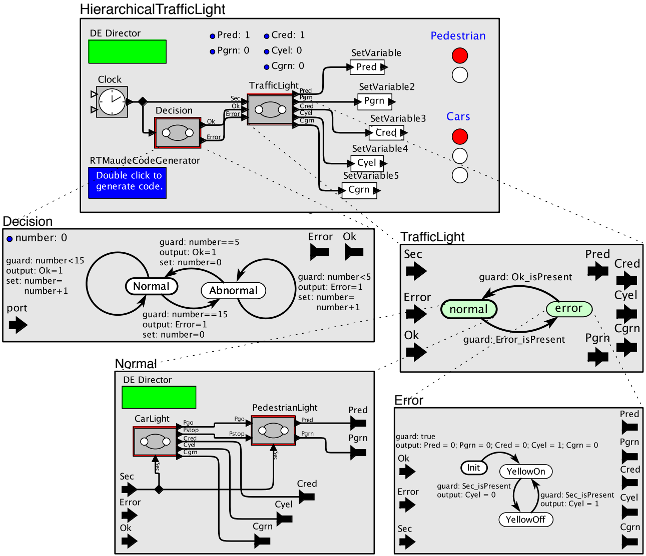
# Context: Real-Time Maude (Ölveczky and Meseguer)

- Expressiveness and modeling convenience
  - algebraic specification!
- User-defined data types; flexible communication; unbounded data structures; dynamic object/message creation/deletion; hierarchical states; ...

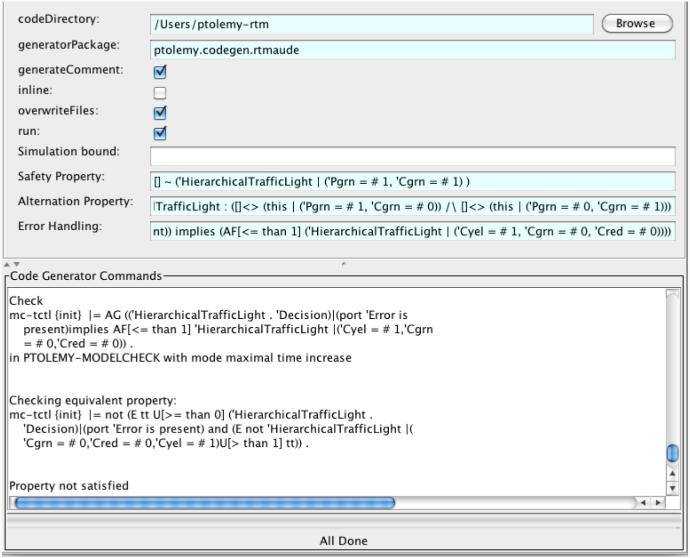
## Context: Real-Time Maude (Ölveczky and Meseguer)

- Expressiveness and modeling convenience
  - algebraic specification!
- User-defined data types; flexible communication; unbounded data structures; dynamic object/message creation/deletion; hierarchical states; ...
- Found unknown flaws in
  - scheduling with reuse of unused budgets (unbounded queues)
  - wireless sensor algorithms (communication, areas, angles)
  - MANET protocol (mobility + communication delay)
  - 50-page multicast protocol (laaarge functions; communication)
  - cloud-based transaction systems
  - cars in Japan
  - avionics systems (complex dynamics; LTL model checking)
  - ...

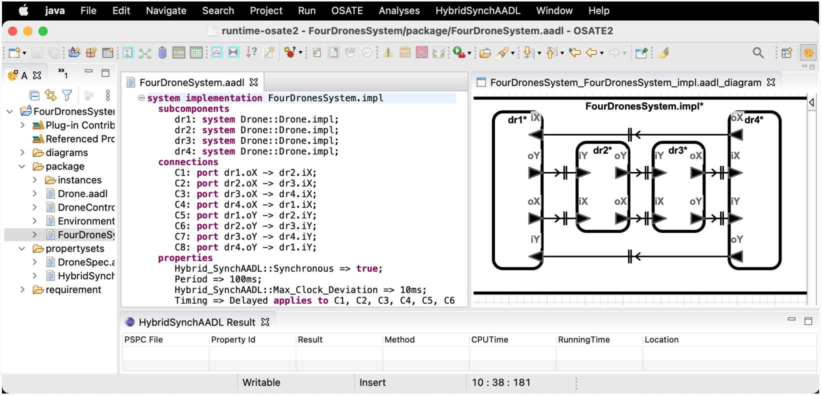
## Ptolemy II DE Models:



Ptolemy II DE Models:

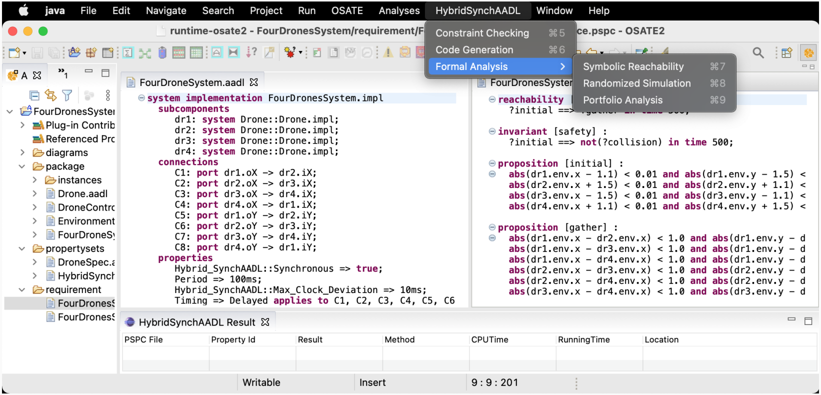


## (Synchronous) AADL:



# Semantics and Analysis for Modeling Languages

(Synchronous) AADL:



## Rewriting Logic and Maude (Meseguer)

### Rewriting Logic

- static parts: algebraic equational specification
- dynamic parts: rewrite rules

```
crl [l] :  t => u if cond .
```

# Rewriting Logic and Maude (Meseguer)

## Rewriting Logic

- static parts: algebraic equational specification
- dynamic parts: rewrite rules

```
crl [l] :  t => u if cond .
```

## Maude: Language and Analysis for Rewriting Logic

- distributed systems in OO style
- simulation
- (explicit-state) search, LTL model checking, ...
- meta-programming
- unification and narrowing

# Maude with SMT Solving

Recent connection to SMT solvers: Yices2, CVC4, Z3

```
check t .

smt-search  t =>* u such that ... . --- undocumented!

op metaCheck : Module Term ~> Bool [special (...)] .
                  --- meta-level
```

# Maude with SMT Solving "in Practice"

- Topmost rules (grab whole state)
- Symbolic "terms/states"

$$\Phi(x, y, ...) \parallel t(x, y, ...)$$

# Maude with SMT Solving "in Practice"

- Topmost rules (grab whole state)
- Symbolic "terms/states"

$$\Phi(x, y, ...) \ || \ t(x, y, ...)$$

- Rule $l : q \longrightarrow r$ **if** $\psi$ turned into rule

$$l : \ \texttt{PHI} \ || \ q \ \longrightarrow \ (\texttt{PHI} \ \textit{and} \ \psi) \ || \ r \ \ \textbf{if} \ \textit{smtCheck}(\texttt{PHI} \ \textit{and} \ \psi)$$

# Real-Time Maude (Ölveczky and Meseguer)

### Real-Time Rewrite Theories

- equational specification for data types
- rewrite rules for instantaneous change
- tick rules

  `crl [l] : {t} => {u} in time τ if ...`

  model time advance

# Real-Time Maude (Ölveczky and Meseguer)

## Real-Time Rewrite Theories

- equational specification for data types
- rewrite rules for instantaneous change
- tick rules

  `crl` [$l$] : {$t$} => {$u$} `in time` $\tau$ `if ...`

  model time advance

## Real-Time Maude

- explicit-state simulation, reachability analysis, ...
- unbounded and time-bounded analysis
- timed CTL model checking

## Explicit-State Analysis in Real-Time Maude

- Tick rules often

  crl [*tick*] : {*t*} => {*f*(*t*, *x*)} in time *x* if *x* <= *mte*(*t*) .

  - *x* new variable
  - non-executable

# Explicit-State Analysis in Real-Time Maude

- Tick rules often

  ```
  crl [tick] :  {t} => {f(t,x)} in time x  if x <= mte(t) .
  ```

  - $x$ new variable
  - non-executable
- Real-Time Maude: explicit-state analysis

# Explicit-State Analysis in Real-Time Maude

- Tick rules often

  ```
  crl [tick] :  {t} => {f(t,x)} in time x  if x <= mte(t) .
  ```

  - $x$ new variable
  - non-executable
- Real-Time Maude: explicit-state analysis
- Real-Time Maude: sample certain moments in time
  - all times not visited
  - analysis unsound/incomplete
  - “counterexamples” real

# Explicit-State Analysis in Real-Time Maude

- Tick rules often

  crl [*tick*] : {*t*} => {*f*(*t*, *x*)} in time *x* if *x* <= *mte*(*t*) .

  - *x* new variable
  - non-executable
- Real-Time Maude: explicit-state analysis
- Real-Time Maude: sample certain moments in time
  - all times not visited
  - analysis unsound/incomplete
  - "counterexamples" real

Real-Time Maude: expressive, but "unsound"

# Goal

# Current Research: Symbolic Real-Time Maude

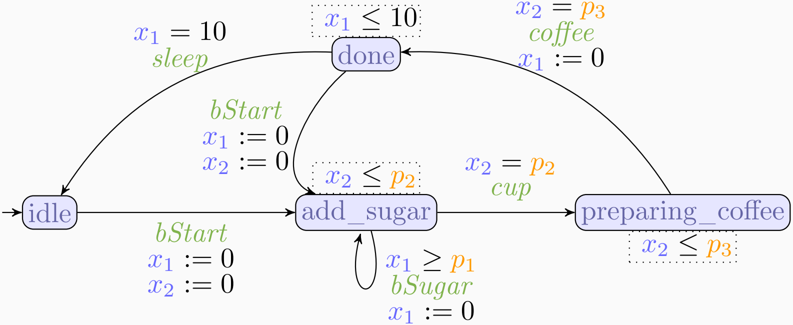
**Research Goals**

- Sound and complete analysis methods for systems beyond timed automata, ...
- Efficient rewriting-with-SMT methods for real-time systems

- Time increase encoded symbolically

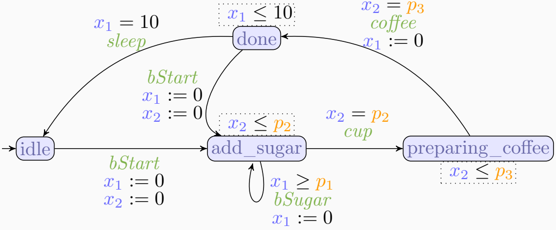
# Maude-with-SMT for Parametric Timed Automata

# Parametric Timed Automata (PTA)




Imitator: state-of-the-art PTA parameter synthesis tool (É. André)

# Representing PTA in Maude



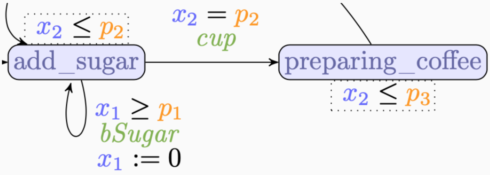

```
mod PTA-COFFEE is pr REAL .
 sorts State NState DState Location .
 subsorts NState DState < State .
 vars X1 X2 P1 P2 P3 T  : Real .

 ops idle addSugar preparingCoffee done : -> Location .

 op <_:_;_> <_;_;_> : Location Real Real Real Real Real -> DState .

 op [_:_;_] <_;_;_> : Location Real ... -> NState .
```

## Representing PTA in Maude (cont.)




```
crl [addSugar-cup] :
    < addSugar : X1 ; X2 >          < P1 ; P2 ; P3 >
   =>
    [ preparingCoffee : X1 ; X2 ]  < P1 ; P2 ; P3 >
  if (X2 === P2 and X2 <= P3) = true .

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

## Analysis with Maude's `smt-search`

Is there a parameter valuation, from initial state with same clock values, such that we reach state done?

```
smt-search  [ idle : X1  ; X2  ] < P1 ; P2 ; P3 >  =>*
            < done : X1' ; X2' > < P1 ; P2 ; P3 >
    such that (X1 === X2 and X1 >= 0/1 and P1 >= 0/1
               and P2 >= 0/1 and P3 >= 0/1) = true .
```

## Analysis with Maude's `smt-search`

Is there a parameter valuation, from initial state with same clock values, such that we reach state done?

```
smt-search  [ idle : X1  ; X2  ] < P1 ; P2 ; P3 >  =>*
            < done : X1' ; X2' > < P1 ; P2 ; P3 >
    such that (X1 === X2 and X1 >= 0/1 and P1 >= 0/1
               and P2 >= 0/1 and P3 >= 0/1) = true .
```

```
Solution 1
state: < done : #3-T ; #1-T + #2-T + #3-T > <P1 ; P2 ; P3>

where X1 === X2 and X1 >= 0/1 and P1 >= 0/1 and P2 >= 0/1 and
  ... and #1-T:Real + #2-T:Real === P3 and
  ... and #3-T:Real <= 10/1 and #1-T:Real + #2-T:Real === P3
```

At least `X2' >= P3` ...

# Reachability

## Maude's `smt-search`

- stops exploring when same symbolic state seen
- tick application adds fresh variable to state
  - search for unreachable states does not terminate

# Reachability

**Maude's `smt-search`**

- stops exploring when same symbolic state seen
- tick application adds fresh variable to state
  - search for unreachable states does not terminate

**Our `reachability` Command**

- stop when symbolic state subsumed by earlier symbolic state
- $\phi_u \;||\; t_u \;\sqsubseteq\; \phi_v \;||\; t_v$
  iff exists $\sigma$ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$

# Reachability

## Maude's `smt-search`

- stops exploring when same symbolic state seen
- tick application adds fresh variable to state
  - search for unreachable states does not terminate

## Our `reachability` Command

- stop when symbolic state subsumed by earlier symbolic state
- $\phi_u \mid\mid t_u \sqsubseteq \phi_v \mid\mid t_v$
  iff exists $\sigma$ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$
- defined subsumption-based `reachability` command
  - Maude's meta-level (`metaMatch` and `metaCheck`)

# Reachability

**Maude's `smt-search`**

- stops exploring when same symbolic state seen
- tick application adds fresh variable to state
  - search for unreachable states does not terminate

**Our `reachability` Command**

- stop when symbolic state subsumed by earlier symbolic state
- $\phi_u \mid\mid t_u \sqsubseteq \phi_v \mid\mid t_v$
  iff exists $\sigma$ such that $t_u = t_v\sigma$ and $\phi_u \implies \phi_v\sigma$
- defined subsumption-based `reachability` command
  - Maude's meta-level (`metaMatch` and `metaCheck`)

**Theorem**

*`reachability` terminates when parametric zone graph of PTA finite*

- *+ soundness/completeness results*

# Analysis and Parameter Synthesis

- $EF\phi$-synthesis and $AG\neg\phi$-synthesis
- $\phi$ involves locations and clocks
- parameters: $\exists X.\varphi$ of resulting constraint $\varphi$
  - quantifier elimination (Z3)
  - done reachable when $p_2 \leq p_3$

# Analysis and Parameter Synthesis

- $EF\phi$-synthesis and $AG\neg\phi$-synthesis
- $\phi$ involves locations and clocks
- parameters: $\exists X.\varphi$ of resulting constraint $\varphi$
  - quantifier elimination (Z3)
  - done reachable when $p_2 \leq p_3$
- Analysis with Maude strategies

**Example**

Can we reach done when some sugar required? (If $p_1 \leq p_2 \leq p_3$)

# Comparied to Imitator

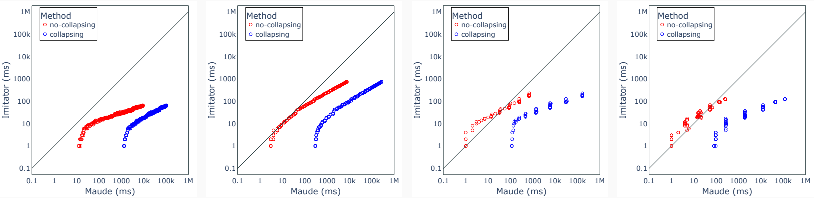
- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
  - not robustness and liveness

# Comparied to Imitator

- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
  - not robustness and liveness
- More than Imitator
  - analysis/parameter synthesis with execution strategies
  - clocks and parameters in propositions

# Comparied to Imitator

- Terminates when Imitator (without heuristics) terminates
- Do most things Imitator can do
  - not robustness and liveness
- More than Imitator
  - analysis/parameter synthesis with execution strategies
  - clocks and parameters in propositions
- Benchmarking on PTA library



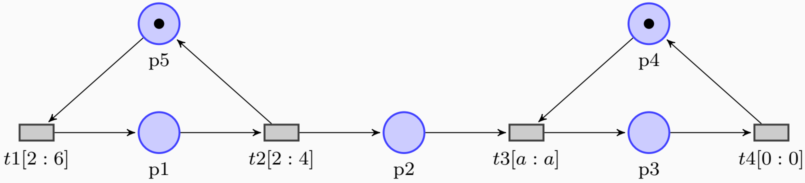

Unfair comparison!

# PTA in Maude-SMT

- Promising
- New `reachability` method
- Parameter synthesis
- Very limited model
  - atomic states
  - no equations
  - linear constraints

## Parametric Interval Time Petri Nets with Inhibitor Arcs (PITPNs)

# Parametric Interval Time Petri Nets with Inhibitor Arcs (PITPNs)




- Unbounded states
- "User-defined" functions
- "Interpreter"

# Representing Nets in Maude

Markings represented `p |-> 2 ; q |-> 3` instead of `p p q q q`

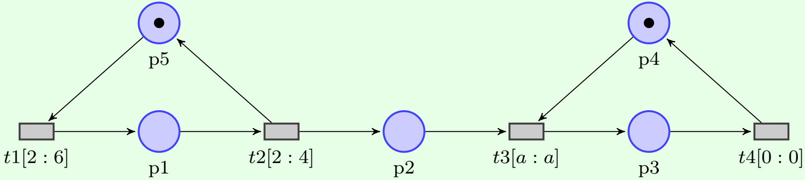
- PITPNs: interleaving semantics (!)
- enables parametric markings with SMT

# Representing Nets in Maude

Markings represented `p |-> 2 ; q |-> 3` instead of `p p q q q`

- PITPNs: interleaving semantics (!)
- enables parametric markings with SMT

**Example**




```
"t1" :  "p5" |-> 1  -->  "p1" |-> 1  in [2 : 6] ;
"t2" :  "p1" |-> 1  -->  "p2" |-> 1 ; "p5" |-> 1  in [2 : 4] ;
"t3" :  "p2" |-> 1 ; "p4" |-> 1  -->  "p3" |-> 1  in [a : a] ;
"t4" :  "p3" |-> 1  -->  "p4" |-> 1  in [0 : 0]
```

## Symbolic Rewrite Semantics: Advance Time

```
var T : Real .

crl [tick] :  tickOk    : M : CLOCKS     : NET
          =>  tickNotOk : M : increaseClocks(M, CLOCKS, NET, T) : NET
if (T >= 0/1 and mte(M, CLOCKS, NET, T)) = true .


op mte : Marking ClockValues Net Real -> Boolean .
eq mte(M, empty, NET, T) = true .
eq mte(M, (L -> R1) ; CLOCKS, (L : PRE --> ... in [T1 : inf]) ; NET, T)
 = mte(M, CLOCKS, NET, T) .
eq mte(M, (L -> R1) ; CLOCKS, (L : PRE --> ... in [T1 : T2])  ; NET, T)
 = (active(M, L : ...) ? T <= T2 - R1 : true)
   and mte(M, CLOCKS, NET, T) .
```

# Symbolic Rewrite Semantics: Applying Transition

```
crl [applyTransition] :
    TS : M : ((L -> T) ; CLOCKS) :
    (L : PRE --> POST inhibit INH in INTERVAL) ; NET)
   =>
    tickOk : ((M - PRE) + POST) :  updateClocks(.., M - PRE, ....) :
    (L : PRE --> ... ; NET)
   if (active(...) and (T in INTERVAL)) = true .


eq updateClocks((L' -> R1) ; CLOCKS, INTERM-M, (L' : PRE --> ...); NET)
 = (L ->  PRE <= INTERM-M ? R1 : 0/1) ; updateClocks(...) .
```

# 1-Unsafe Nets Reachable?

```
smt-search   tickOk :  m : initClocks(net) : net
       =>*   TICK   : M   : CLOCKS          : NET
  such that (a:Real >= 0/1 and not k-safe(1, M)) = true .
```

Unsafe net reachable if $a \geq 4$

# Problems – And Solutions

- Subsumption $T \sqsubseteq U$ not sufficient
    - $[\![T]\!] \subseteq [\![U]\!]$ does not imply $T \sqsubseteq U$

# Problems – And Solutions

- Subsumption $T \sqsubseteq U$ not sufficient
    - $[\![T]\!] \subseteq [\![U]\!]$ does not imply $T \sqsubseteq U$
- Defined new subsumption $\preceq$
    - $T \preceq U$ iff $[\![T]\!] \subseteq [\![U]\!]$ for PITPNs

# Problems – And Solutions

- Subsumption $T \sqsubseteq U$ not sufficient
  - $[\![T]\!] \subseteq [\![U]\!]$ does not imply $T \sqsubseteq U$
- Defined new subsumption $\preceq$
  - $T \preceq U$ iff $[\![T]\!] \subseteq [\![U]\!]$ for PITPNs

**Theorem**

*Reachability with $\preceq$-folding terminates when symbolic state space finite!*

# Reachability Analyses with New Folding

1. Maude search: carry symbolic states visited in current behavior
   - stop when current state subsumed by previous symbolic state in same branch
2. Store all visited symbolic states + implement search analysis
   - encode breadth-first search at Maude meta-level

# Reachability Analyses with New Folding

1. Maude search: carry symbolic states visited in current behavior
   - stop when current state subsumed by previous symbolic state in same branch
2. Store all visited symbolic states + implement search analysis
   - encode breadth-first search at Maude meta-level

**Example**

```
search init(net, m0, a < 4 ...)
        =>*   S : φ' || ( TICK : M : CLOCKS : NET )
                such that smtCheck(φ' and not k-safe(1,M)) .
```

returns `No solution`

# EF-Synthesis

For what parameter values is the net not 1-safe?

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

# EF-Synthesis

For what parameter values is the net not 1-safe?

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

- quantifier elimination of resulting constraint gives desired values of parameters
- $a \geq 4$

# Initial Marking Synthesis

$m_s$ parametric initial marking; $0 \leq x_i \leq 1$ tokens in place $p_i$

```
safety-syn(net, m_s, φ_0, k-safe(1,M))
```

- net 1-safe when $x_1 = x_3 = 0$ and $0 \leq x_2 \leq 1$

# More Analysis

- Analysis with user-defined strategies
- Non-nested temporal properties
- ...

# Comparison with Roméo

- Terminates when Roméo terminates
  - not vice versa

## Comparison with Roméo

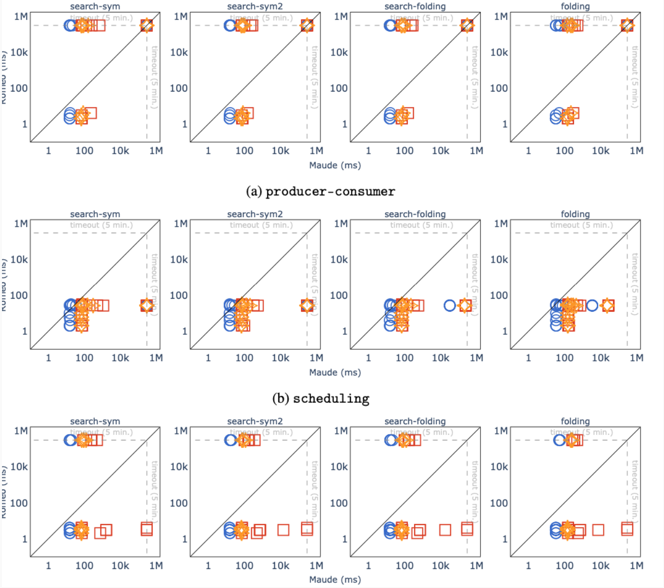
- Terminates when Roméo terminates
  - not vice versa
- Do almost everything Roméo can do
  - not parameters in temporal property time bounds

# Comparison with Roméo

- Terminates when Roméo terminates
    - not vice versa
- Do almost everything Roméo can do
    - not parameters in temporal property time bounds
- More:
    - parametric/synthesize initial markings
    - user-defined strategies (restricting possible behaviors)
    - ...

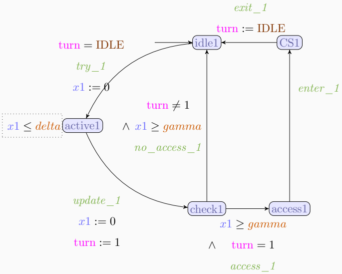
# Benchmarking

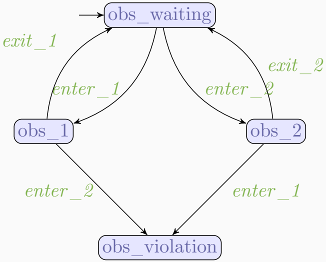
Comparison with Roméo; different implementations and solvers



# Networks of Parametric Timed Automata with Variables

# Networks of Parametric Timed Automata with Variables (NPTAVs)




## "Interpreter" for NPTAVs

- sound/correct
- folding-based analysis
- terminates whenever PZG finite

# Parameter Synthesis: Fischer

**Example**

```
Maude> red synthesis-folding  in 'FISCHER :
             init => (observer @ obs-violation) .
```

# Parameter Synthesis: Fischer

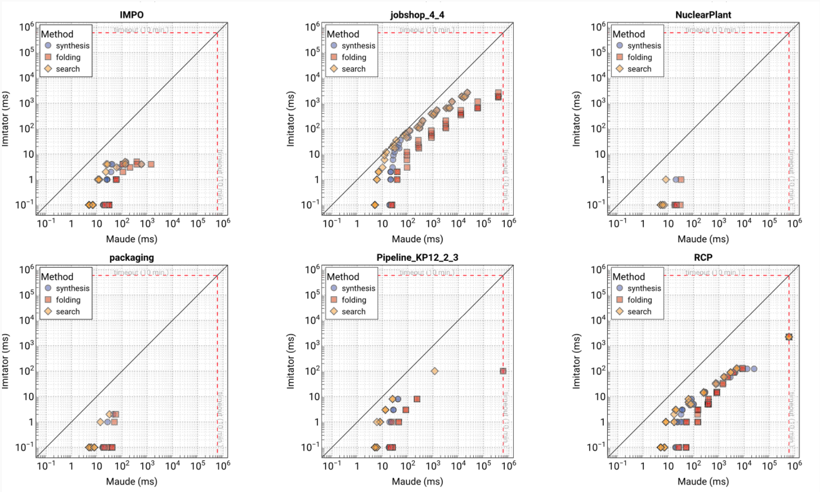
**Example**

```
Maude> red synthesis-folding  in 'FISCHER :
              init => (observer @ obs-violation) .

result ConjRelLinRExpr: rr(var(gamma)) + -1 * rr(var(delta)) <= 0
       and -1 * rr(var(delta)) <= 0 and  -1 * rr(var(gamma)) <= 0
```

Bad states reachable when $\gamma \leq \delta$

# Benchmarking: Imitator vs Maude-SMT



# Concluding Remarks

## Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- New folding: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...

# Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- New folding: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...
- Can do essentially everything Imitator/Roméo can do, and ...
  - analysis with strategies
  - parametric markings
  - LTL model checking
  - clocks and parameters in "propositions"

# Concluding Remarks I

- Semantics/interpreter for NPTAVs and PITPNs
- Symbolic states (Maude-with-SMT analysis)
- New folding: termination when Imitator/Roméo terminates
- Reachability, parameter synthesis, ...
- Can do essentially everything Imitator/Roméo can do, and ...
  - analysis with strategies
  - parametric markings
  - LTL model checking
  - clocks and parameters in "propositions"
- Benchmarking
  - worse than Imitator
  - comparable with Roméo (?)

# Concluding Remarks II

- Methods towards "Symbolic Real-Time Maude"
- Maude-with-SMT for not-entirely-trivial systems
- Different subsumption relations; different reachability analysis methods; Fourier-Motzkin quantifier elimination; parameter synthesis; ...
- Extend to classes of systems beyond automata and Petri nets
  - linear arithmetic/reals?
  - topmost rules?
  - ...
- Symbolic full (timed) temporal logic model checking