Specification and Analysis in Real-Time Maude

Peter Ölveczky

University of Oslo

Based on joint work with José Meseguer, Erika Ábrahám, Daniela Lepri, and many others

Content

Background: Rewriting Logic and Maude

- Rewriting logic [Meseguer'90]
  - data types defined by algebraic equational specifications
  - dynamic behaviors defined by rewrite rules

$$l: \quad t \longrightarrow u \textbf{ if } cond$$

Background: Rewriting Logic and Maude

- Rewriting logic [Meseguer'90]
  - data types defined by algebraic equational specifications
  - dynamic behaviors defined by rewrite rules

$$l : \quad t \longrightarrow u \ \textbf{if} \ cond$$

- Maude: language and tool for rewriting logic
  - simulation
  - reachability analysis
  - LTL model checking
  - ...

Background: Rewriting Logic and Maude (II)

Rewriting logic:

- expressive and general . . .
- . . . yet simple and intuitive
  - simple model of concurrent objects
  - different forms of communication easily defined
  - any computable data type definable
  - . . .

Extend to Real-Time Systems

How to extend Maude to real-time systems?

Real-Time Maude Ölveczky & Meseguer

- Time advance modeled by tick rewrite rules

  `crl [`l`] : {`t`} => {`t'`} in time `τ` if `*cond*

  - global state has form $\{t\}$

Real-Time Maude Ölveczky & Meseguer

- Time advance modeled by tick rewrite rules

  `crl [`l`] : {`t`} => {`t'`} in time` τ `if` *cond*

  - global state has form {t}
- "Ordinary" rewrite rules model instantaneous change

Specifying OO Real-Time Systems

Tick rule for OO systems:

```
var τ : Time .

crl [l] : {t}  => {timeEffect(t, τ)}  in time τ if τ <= mte(t)
```

Example: "Retrograde" Clock

- state: {clock(*r*)} or {stopped-clock(*r*)}
- dense time domain
- clock can stop at any time
- retrograde clock: clock(24) must be reset to clock(0)

Real-Time Maude Specification

```
(tmod DENSE-CLOCK is pr POSRAT-TIME-DOMAIN .
  ops clock stopped-clock : Time -> System .
  vars R R' : Time .

  crl [tickWhenRunning] :
      {clock(R)} => {clock(R + R')} in time R'
        if  R' <= 24 - R .

  rl [tickWhenStopped] :
     {stopped-clock(R)} => {stopped-clock(R)} in time R' .

  rl [reset] :   clock(24) => clock(0) .

  rl [batteryDies] :  clock(R) => stopped-clock(R) .
endtm)
```

Main Challenge

How to deal with dense time?

Time Sampling

- Tick rules "cover" dense time domain
  - not executable

Time Sampling

- Tick rules "cover" dense time domain
  - not executable
- "On-the-fly discretization:" time sampling strategies
  - advance time by default value Δ
  - advance time as much as possible ("event-driven simulation")

Time Sampling

- Tick rules "cover" dense time domain
  - not executable
- "On-the-fly discretization:" time sampling strategies
  - advance time by default value Δ
  - advance time as much as possible ("event-driven simulation")
- Analysis not sound/complete: all behaviors not covered

Real-Time Maude Analysis

- Timed rewriting
  - simulate system to time T
- Timed reachability analysis
- LTL model checking
  - unbounded/time-bounded
  - clocked/un-clocked
- Timed CTL model checking

Reachability Analysis

Define time sampling:

```
Maude> (set tick def 1 .)
```

- analysis w.r.t. this strategy
- Can {clock(25)} be reached?

```
(utsearch [1] {clock(0)} =>* {clock(25)} .)
```

Reachability Analysis

Define time sampling:

```
Maude> (set tick def 1 .)
```

- analysis w.r.t. this strategy
- Can `{clock(25)}` be reached?

  ```
  (utsearch [1] {clock(0)} =>* {clock(25)} .)
  ```

- State `{clock(1/2)}` not found:

  ```
  (utsearch [1] {clock(0)} =>* {clock(1/2)} .)
  ```

In Context (I)

- Timed automata
  - restricted formalism ...
  - ... many properties decidable
  - state-of-the-art tools: Uppaal, RED
- Time(d) Petri nets
  - fixed model of comminication

In Context (I)

- Timed automata
  - restricted formalism ...
  - ... many properties decidable
  - state-of-the-art tools: Uppaal, RED
- Time(d) Petri nets
  - fixed model of comminication
- IF, TE-LOTOS, etc:
  - separate formalisms for data types, dynamic behavior, and time
  - based on fixed communication primitives
- Moby/RT
  - designs specified as PLC-automata
  - translated into timed automata for model checking
- BIP (Behavior, Interaction, Priority)
  - "Behavior is described as a Petri net extended with data and functions described in C"

In Context (II)

Real-Time Maude:

- simple and intuitive
- expressive
- any data type
- unbounded data structures
- dynamic object/message creation/deletion
- hierarchical structures
- easy to define communication forms

In Context (II)

Real-Time Maude:

- simple and intuitive
- expressive
- any data type
- unbounded data structures
- dynamic object/message creation/deletion
- hierarchical structures
- easy to define communication forms

- properties in general undecidable
- discrete abstraction may not exist in general

Outline

Real-Time Maude

Analysis

Applications
- "Concrete" Systems
- Formal Semantics and Analysis for MDE Languages

Soundness/Completeness

- Real-Time Maude analyses "incomplete" for dense time
  - formalism too general for "region graphs"

Soundness/Completeness

- Real-Time Maude analyses "incomplete" for dense time
  - formalism too general for "region graphs"
- Can we have sound/complete maximal time sampling analysis?

Sound/Complete Untimed Analysis [Ölveczky–Meseguer'06]

Time-robust theories:

- "well-behaved" timed behavior
- no instantaneous actions between maximal ticks (that ...)

Sound/Complete Untimed Analysis [Ölveczky–Meseguer'06]

Time-robust theories:

- "well-behaved" timed behavior
- no instantaneous actions between maximal ticks (that ...)
- Conditions for OO specifications:
  - $\mathtt{mte}(\mathtt{timeEffect}(t, r)) = \mathtt{mte}(t) \dot{-} r$, for all $r \leq \mathtt{mte}(t)$.
  - $\mathtt{timeEffect}(t, 0) = t$.
  - $\mathtt{timeEffect}(\mathtt{timeEffect}(t, r), r') = \mathtt{timeEffect}(t, r + r')$, for $r + r' \leq \mathtt{mte}(t)$.
  - $\mathtt{mte}(\sigma(l)) = 0$ for each ground instance $\sigma(l)$ of a left-hand side of an instantaneous rewrite rule.

Ticks and Propositions

- Atomic propositions P tick-stabilizing
  - valuation of set of propositions P changes at most once in any sequence of ticks between two maximal tick steps

Ticks and Propositions

- Atomic propositions P tick-stabilizing
  - valuation of set of propositions P changes at most once in any sequence of ticks between two maximal tick steps
- P tick-invariant
  - P unchanged by applying tick rules

Sound/Complete Untimed Analysis (II)

- Analysis with maximal time sampling satisfies the same $LTL \setminus \{\bigcirc\}$ formulas as the timed fair paths in $\mathcal{R}$ if
  - $\mathcal{R}$ is time-robust
  - P tick-stabilizing

$$\mathcal{R}, L_P, t_0 \models^{tf} \Phi \qquad \text{if and only if} \qquad \mathcal{R}^{maxDef(r),nz}, L_P, t_0 \models \Phi.$$

Sound/Complete Untimed Analysis (II)

- Analysis with maximal time sampling satisfies the same $LTL \setminus \{\bigcirc\}$ formulas as the timed fair paths in $\mathcal{R}$ if
    - $\mathcal{R}$ is time-robust
    - P tick-stabilizing

$$\mathcal{R}, L_P, t_0 \models^{tf} \Phi \qquad \text{if and only if} \qquad \mathcal{R}^{maxDef(r),nz}, L_P, t_0 \models \Phi.$$

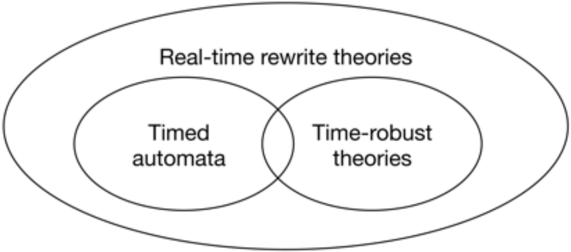
- Holds for time-bounded model checking if $\mathcal{R}$ time-robust and P tick-invariant

Sound/Complete Untimed Analysis (II)

- Analysis with maximal time sampling satisfies the same $LTL \setminus \{\bigcirc\}$ formulas as the timed fair paths in $\mathcal{R}$ if
  - $\mathcal{R}$ is time-robust
  - P tick-stabilizing

$$\mathcal{R}, L_P, t_0 \models^{tf} \Phi \qquad \text{if and only if} \qquad \mathcal{R}^{maxDef(r),nz}, L_P, t_0 \models \Phi.$$

- Holds for time-bounded model checking if $\mathcal{R}$ time-robust and P tick-invariant



Timed Temporal Logic

- So far: untimed LTL model checking
  - “the airbag must eventually deploy after crash detected”
  - “BO eventually closes G”

Timed Temporal Logic

- So far: untimed LTL model checking
  - "the airbag must eventually deploy after crash detected"
  - "BO eventually closes G"
- Timed temporal logics
  - "the airbag must deploy within 10ms after crash"
  - "BO closes G within one year of inauguration"

Real-Time Maude's TCTL Model Checker

- Explicit-state timed CTL model checker for Real-Time Maude
- TCTL: temporal operators with time intervals: $\exists\, \phi\, \mathcal{U}_{[r_1, r_2]}\, \phi'$
  - $\forall\Box\,(\mathit{crash} \implies \forall\Diamond_{\leq 10ms}\ \mathit{airbagDeployed})$
  - $\forall\Box\,((\mathit{inauguration}(BO) \wedge \mathit{open}(G)) \implies \forall\Diamond_{\leq one\ year}\ \mathit{closed}(G))$

D. Lepri, E. Ábrahám, P.C. Ölveczky: Sound and complete timed CTL model checking of timed Kripke structures and real-time rewrite theories. Science of Computer Programming 99 (2015)

Intended Semantics

What is the intended semantics of a Real-Time Maude model?

$\{clock(R)\} \rightarrow \{clock(R + R')\}$ **in time** R' **if** $R' \leq 24 - R$

Intended Semantics

What is the intended semantics of a Real-Time Maude model?

$\{clock(R)\} \rightarrow \{clock(R + R')\}$ **in time** R' **if** $R' \leq 24 - R$

- Should $\forall \Diamond_{[1,2]}$ `True` hold from $\{clock(0)\}$?

Intended Semantics

What is the intended semantics of a Real-Time Maude model?

$\{clock(R)\} \rightarrow \{clock(R + R')\}$ **in time** R' **if** $R' \leq 24 - R$

- Should $\forall \Diamond_{[1,2]}$ `True` hold from $\{clock(0)\}$?
- Pointwise semantics
  - only visited states into account
  - $\forall \Diamond_{[1,2]}$ `True` does not hold from $\{clock(0)\}$

Intended Semantics

What is the intended semantics of a Real-Time Maude model?

$\{clock(R)\} \rightarrow \{clock(R + R')\}$ **in time** R' **if** $R' \leq 24 - R$

- Should $\forall \diamond_{[1,2]}$ `True` hold from $\{clock(0)\}$?
- Pointwise semantics
  - only visited states into account
  - $\forall \diamond_{[1,2]}$ `True` does not hold from $\{clock(0)\}$
- Continuous semantics
  - tick rule interpreted as representing continuous process
  - $\forall \diamond_{[1,2]}$ `True` holds from $\{clock(0)\}$

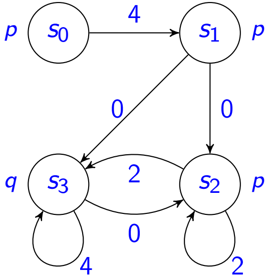
Soundness and Completeness

Soundness and completeness for maximal time sampling analyses of untimed TL do not carry over to timed CTL

- maximal time sampling analysis does not satisfy $\exists \Diamond_{[1,2]}$ `True`
- ... or $\forall \Diamond_{[1,2]}$ `True`

From Continuous to Pointwise

- Reduce model checking under continuous semantics to pointwise case
- For timed Kripke structures



From Continuous to Pointwise for Timed Kripke Structures

- Assume
  - dense time (abstract axiomatization of time)
  - tick-invariance

From Continuous to Pointwise for Timed Kripke Structures

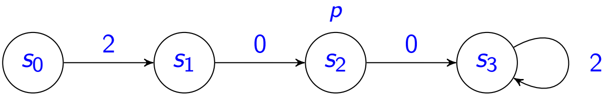
- Assume
  - dense time (abstract axiomatization of time)
  - tick-invariance
- Idea: stop time advance "when something could happen"
- Dense time: γ is the *gcd* of
  - any non-zero time value in the TCTL formula
  - any non-zero maximal tick duration

From Continuous to Pointwise for Timed Kripke Structures (II)

Advancing time by γ ($= gcd(durations, formulaBounds)$) is not sufficient

From Continuous to Pointwise for Timed Kripke Structures (II)

Advancing time by γ ($= gcd(durations, formulaBounds)$) is not sufficient

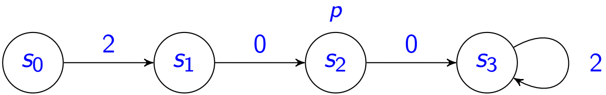


- pointwise behavior

$$\pi = \neg p \xrightarrow{2} \neg p \xrightarrow{0} p \xrightarrow{0} \neg p \xrightarrow{2} \neg p \xrightarrow{2} \cdots (\neg p \text{ forever})$$

From Continuous to Pointwise for Timed Kripke Structures (II)

Advancing time by γ $(= gcd(durations, formulaBounds))$ is not sufficient



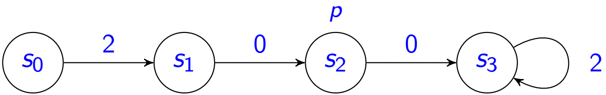
- pointwise behavior

$$\pi = \neg p \xrightarrow{2} \neg p \xrightarrow{0} p \xrightarrow{0} \neg p \xrightarrow{2} \neg p \xrightarrow{2} \cdots (\neg p \text{ forever})$$

- φ is $\exists\, (\exists \Diamond_{=2}\, p)\, \mathcal{U}_{=2}\, \texttt{true}$

From Continuous to Pointwise for Timed Kripke Structures (II)

Advancing time by γ $(= gcd(durations, formulaBounds))$ is not sufficient



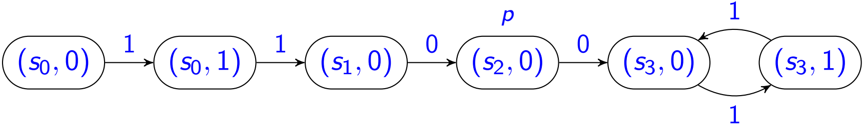
- pointwise behavior

$$\pi = \neg p \xrightarrow{2} \neg p \xrightarrow{0} p \xrightarrow{0} \neg p \xrightarrow{2} \neg p \xrightarrow{2} \cdots (\neg p \text{ forever})$$

- φ is $\exists\,(\exists\,\Diamond_{=2}\, p)\,\mathcal{U}_{=2}\,\texttt{true}$
- γ is 2: splitting into γ-steps gives no additional runs!
- φ holds in pointwise semantics but not in continuous

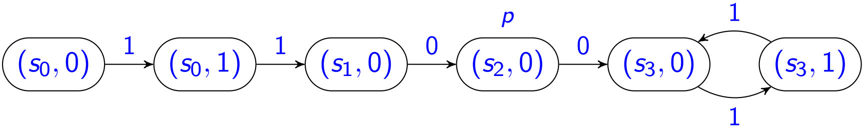
From Continuous to Pointwise for Timed Kripke Structures (III)

- Solution: split each step into steps of length $\gamma/2$
- Previous system split into:



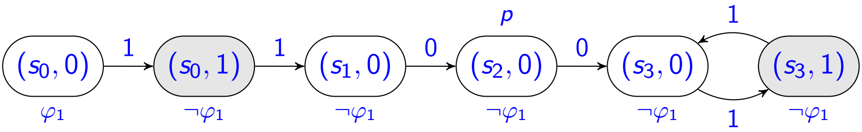
From Continuous to Pointwise for Timed Kripke Structures (III)

- Solution: split each step into steps of length $\gamma/2$
- Previous system split into:

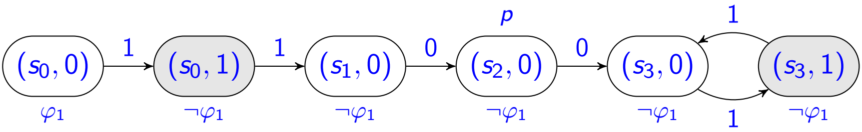


- $\exists\, (\exists\, \Diamond_{=2}\, p)\, \mathcal{U}_{=2}$ `true` does not hold here!

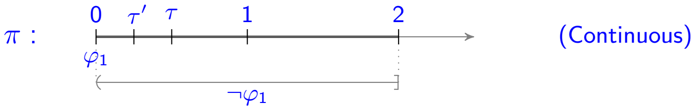
One More Subtlety



One More Subtlety



- φ_1 is $\exists\Diamond_{=2}\ p$
- $\exists\ \varphi_1\ \mathcal{U}\ \neg\varphi_1$ does not hold in continuous semantics



From Continuous to Pointwise for Timed Kripke Structures (IV)

Main result:

$$\mathcal{TK}, s, \models_{cont} \varphi \iff \mathcal{TK}_a^{\gamma/2}, (s, 0) \models_{pointwise} \beta(\alpha(\varphi))$$

- α transforms formula to one with closed intervals
- β transforms formula to solve previous slide

Model Checking Timed Kripke Structures

Model checking timed Kripke structures:

- Extends and optimizes algorithm by Laroussinie, Markey, and Schoebelen
    - formula into normal form
    - recursively compute sets of states sastifying subformulas

TCTL Model Checking for Real-Time Maude

Sound and complete TCTL model checking using maximal time sampling and the $\gamma/2$-transformation:

$$\mathcal{TK}_t(\mathcal{R}^{max}, AP)_a^{\gamma/2}, (t, 0) \models_{pointwise} \beta(\alpha(\varphi))$$

$$\Updownarrow$$

$$\mathcal{TK}(\mathcal{R}, AP), t \models_{cont} \varphi$$

since

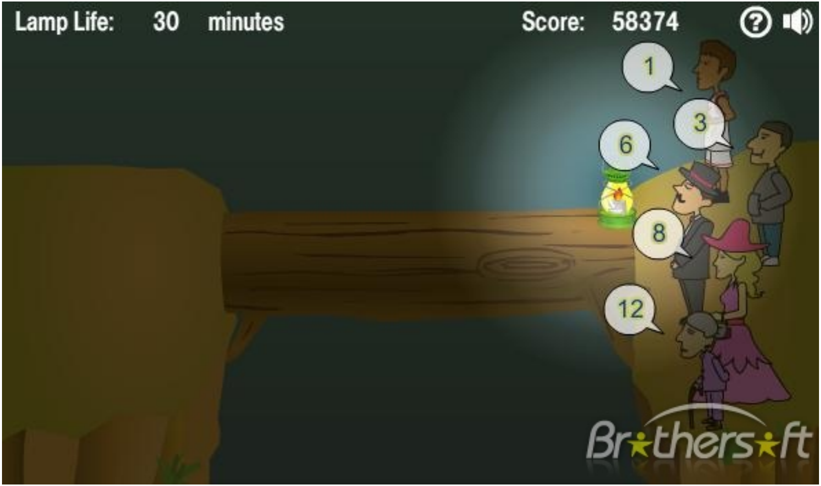
$$\mathcal{TK}(\mathcal{R}^{max}, AP), t \models_{cont} \varphi \iff \mathcal{TK}(\mathcal{R}, AP), t \models_{cont} \varphi$$

when $\mathcal{R}$ time-robust and AP tick-invariant

TCTL Model Checker for Real-Time Maude

- With/without $\gamma/2$-transformation
- Implemented in Maude (meta-level)
- No counterexample provided!

Benchmarking: Crossing the Bridge



Benchmarking: Crossing the Bridge

Crossing the Bridge

- Initial state and property

```
eq init(N) = person(5 * N,false)   person(10 * N,false)
             person(20 * N,false)  person(25 * N,false)
             lamp(false) .

op safe : -> Prop .
eq {person(T:Time, false) S:System} |= safe = false .
eq {S:System} |= safe = true [owise] .
```

- Model checking:

```
Maude> (mc-tctl {init(1)} |= AG EF[<= than 85] safe .)
```

Benchmarking

Initial state	TSMV	Real-Time Maude (pointwise)	Real-Time Maude (continuous)	RED 7.0
init(1)	0.074	0.149	1.266	0.429
init(10)	0.148	0.168	0.999	0.408
init(100)	1.443	0.168	1.012	0.404
init(1000)	57.426	0.327	1.014	0.426
init+(2)	0.191	0.746	6.864	1.044
init+(4)	0.280	1.772	17.752	2.153
init+(8)	0.759	5.227	57.580	16.912
init+(12)	1.080	11.198	129.957	79.319
init+(16)	1.515	19.620	233.414	241.098

Execution times for the bridge crossing problem (in seconds).

Outline

Real-Time Maude

Analysis

Applications

- "Concrete" Systems
- Formal Semantics and Analysis for MDE Languages

Main Question

Complex data types; unbounded data structures; flexible communication models; hierarchical objects; dynamic object creation/deletion; . . .

Main Question

Complex data types; unbounded data structures; flexible communication models; hierarchical objects; dynamic object creation/deletion; . . .

Are there systems where Real-Time Maude's expressiveness needed

and

Real-Time Maude analysis yields interesting results?

Classes of Applications

- "Concrete" systems/protocols
- Semantic framework for real-time systems
- Formal analysis tool for other languages
- ...

AER/NCA [with C. Talcott and others]

AER/NCA:

- Multicast for active networks
  - 50 pages of use cases
  - involves link capacity and propagation delay, packet sizes, etc.

AER/NCA [with C. Talcott and others]

AER/NCA:

- Multicast for active networks
  - 50 pages of use cases
  - involves link capacity and propagation delay, packet sizes, etc.
- Real-Time Maude analysis found all known design errors

AER/NCA [with C. Talcott and others]

AER/NCA:

- Multicast for active networks
  - 50 pages of use cases
  - involves link capacity and propagation delay, packet sizes, etc.
- Real-Time Maude analysis found all known design errors
- . . . and additional unknown serious design errors

AER/NCA [with C. Talcott and others]

AER/NCA :

- Multicast for active networks
    - 50 pages of use cases
    - involves link capacity and propagation delay, packet sizes, etc.
- Real-Time Maude analysis found all known design errors
- . . . and additional unknown serious design errors

Key Real-Time Maude features:

- detailed parametric model of communication
- laaaaarge functions
- multiple class inheritance to combine subprotocols

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded
- Search found missed hard deadline

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded
- Search found missed hard deadline
- Extensive "Monte-Carlo simulation" did not find flaw

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded
- Search found missed hard deadline
- Extensive "Monte-Carlo simulation" did not find flaw

CASH Scheduling Algorithm [with M. Caccamo]

CASH: State-of-the-art scheduling algorithm

- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded
- Search found missed hard deadline
- Extensive "Monte-Carlo simulation" did not find flaw

Key Real-Time Maude feature: unbounded data structures

OGDC Wireless Sensor Network Algorithm

[with S. Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension

OGDC Wireless Sensor Network Algorithm

[with S. Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension
- New form of communication: radio transmission
  - easy to specify communication in Real-Time Maude
- Real-Time Maude simulations found unknown major flaw

OGDC Wireless Sensor Network Algorithm

[with S. Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension
- New form of communication: radio transmission
    - easy to specify communication in Real-Time Maude
- Real-Time Maude simulations found unknown major flaw
- Performance estimation as good as WSN simulation tool

OGDC Wireless Sensor Network Algorithm

[with S. Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension
- New form of communication: radio transmission
  - easy to specify communication in Real-Time Maude
- Real-Time Maude simulations found unknown major flaw
- Performance estimation as good as WSN simulation tool

Key Real-Time Maude features:

- easy to define "new" model of communication
- complex data types and functions (areas, angles, distances)
- simulation

Megastore and Megastore-CGC [with J. Grov]

Megastore : Google's distributed data store

Megastore and Megastore-CGC [with J. Grov]

Megastore : Google's distributed data store

- Developed Real-Time Maude specification

Megastore and Megastore-CGC [with J. Grov]

Megastore : Google's distributed data store

- Developed Real-Time Maude specification
- Megastore:
  - consistency for transactions accessing one entity group
- Megastore-CGC:
  - consistency for transactions accessing multiple entity groups

Megastore and Megastore-CGC [with J. Grov]

Megastore: Google's distributed data store

- Developed Real-Time Maude specification
- Megastore:
  - consistency for transactions accessing one entity group
- Megastore-CGC:
  - consistency for transactions accessing multiple entity groups

Key Real-Time Maude features:

- simple and intuitive language
- automatic "testing" highly appreciated
- analysis of performance and correctness

Some Other "Concrete" Applications

- Found several bugs in embedded car software used by major car makers (Japan)
  - bugs not found by model-checking tools employed in industry

Some Other "Concrete" Applications

- Found several bugs in embedded car software used by major car makers (Japan)
  - bugs not found by model-checking tools employed in industry
- ERMTS/ETCS railway signaling and control system
- Leader election for mobile ad hoc networks
- EIGRP Cisco routing protocol (Riesco, Verdejo)
- Parts of NORM multicast protocol developed by IETF

Formal Semantics and Analysis for MDE Languages

- Modeling languages for embedded systems
  - intuitive domain-specific modeling
  - often lack formal semantics and analysis

Formal Semantics and Analysis for MDE Languages

- Modeling languages for embedded systems
  - intuitive domain-specific modeling
  - often lack formal semantics and analysis
- Real-Time Maude semantic framework and formal analysis tool for such languages

Formal Semantics and Analysis for MDE Languages

- Modeling languages for embedded systems
  - intuitive domain-specific modeling
  - often lack formal semantics and analysis
- Real-Time Maude semantic framework and formal analysis tool for such languages
  - modeling languages used in industry
    - Ptolemy II DE models
    - AADL avionics modeling standard (subset)
    - DOCOMO's $\mathcal{L}$ language

Formal Semantics and Analysis for MDE Languages

- Modeling languages for embedded systems
  - intuitive domain-specific modeling
  - often lack formal semantics and analysis
- Real-Time Maude semantic framework and formal analysis tool for such languages
  - modeling languages used in industry
    - Ptolemy II DE models
    - AADL avionics modeling standard (subset)
    - DOCOMO's $\mathcal{L}$ language
  - timed model transformations
    - Real-Time MOMENT-2
    - e-Motions
  - Orc, Timed Rebeca, ...

Ptolemy II DE Models [joint work with Kyungmin Bae et al.]

Ptolemy II

- graphical modeling and simulation tool from UC Berkeley
- hierarchical composition of actors

Ptolemy II DE Models [joint work with Kyungmin Bae et al.]

Ptolemy II

- graphical modeling and simulation tool from UC Berkeley
- hierarchical composition of actors
- different models of computation
- Discrete Event (DE) models:
    - timed
    - fixed-point semantics of synchronous languages

Ptolemy II DE Models [joint work with Kyungmin Bae et al.]

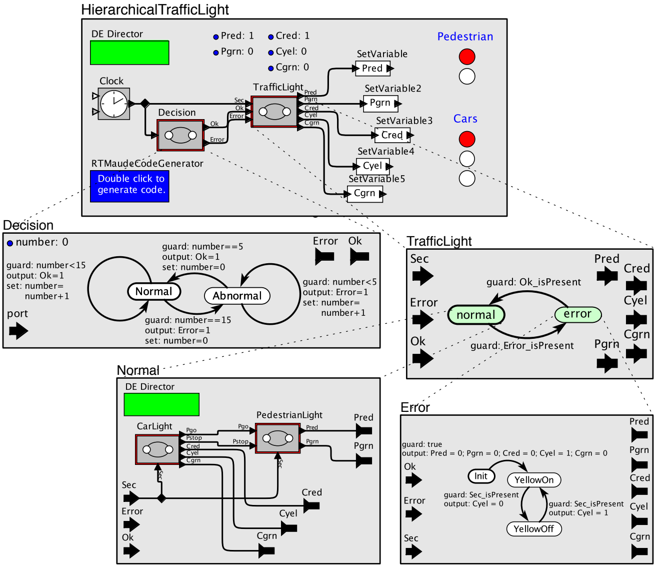
Ptolemy II

- graphical modeling and simulation tool from UC Berkeley
- hierarchical composition of actors
- different models of computation
- Discrete Event (DE) models:
  - timed
  - fixed-point semantics of synchronous languages

Key Maude features:

- hierarchical configurations
- expressiveness
- unbounded data structures
- parametric atomic propositions

Ptolemy II: Fault-Tolerant Traffic Lights



Formal Analysis of Ptolemy DE Models

Predefined parametric propositions:

$$\mathit{actorId} \;\texttt{|}\; \mathit{var}_1 \;=\; \mathit{value}_1, \ldots, \mathit{var}_n \;=\; \mathit{value}_n$$

$$\mathit{actorId} \;\texttt{@}\; \mathit{location}$$

$$\mathit{actorId} \;\texttt{|}\; \texttt{port}\; p \;\texttt{is}\; \mathit{value}$$

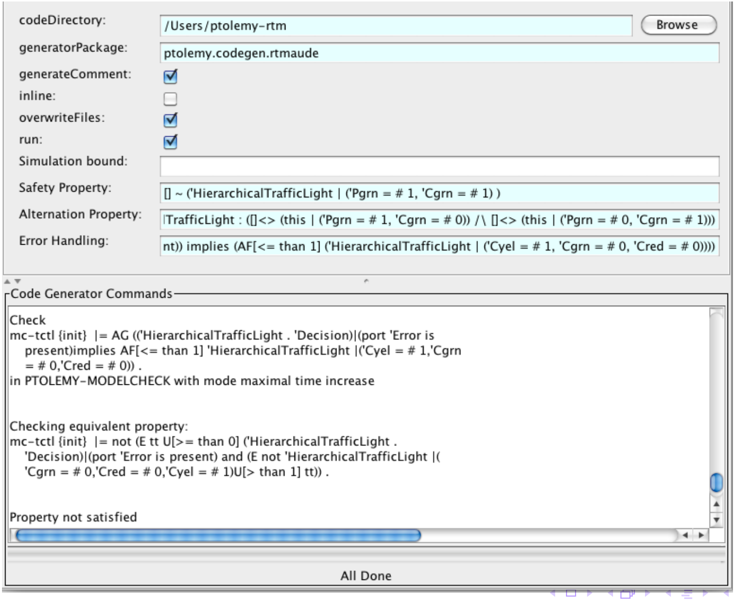
$$\mathit{actorId} \;\texttt{|}\; \texttt{port}\; p \;\texttt{is}\; \mathit{status}$$

A Timed CTL Property

Car light will show only yellow within time 1 of a failure:

```
AG (('HierarchicalTrafficLight . 'Decision |
        port 'Error is present)
   => AF[<= 1] ('HierarchicalTrafficLight |
                    'Cyel = 1, 'Cgrn = 0, 'Cred = 0))
```

Analyzing Ptolemy II Models Within Ptolemy



Concluding Remarks

- Real-Time Maude formalism expressive and intuitive

Concluding Remarks

- Real-Time Maude formalism expressive and intuitive
- Sound/complete timed CTL model checking abstraction/discretization for time-robust theories
    - sound/complete analysis for new classes of systems

Concluding Remarks

- Real-Time Maude formalism expressive and intuitive
- Sound/complete timed CTL model checking abstraction/discretization for time-robust theories
  - sound/complete analysis for new classes of systems
- Used on state-of-the-art systems in different domains
  - value added to domain-specific analysis

Concluding Remarks

- Real-Time Maude formalism expressive and intuitive
- Sound/complete timed CTL model checking abstraction/discretization for time-robust theories
  - sound/complete analysis for new classes of systems
- Used on state-of-the-art systems in different domains
  - value added to domain-specific analysis
- Useful both as simulation tool and model checker

Concluding Remarks

- Real-Time Maude formalism expressive and intuitive
- Sound/complete timed CTL model checking abstraction/discretization for time-robust theories
  - sound/complete analysis for new classes of systems
- Used on state-of-the-art systems in different domains
  - value added to domain-specific analysis
- Useful both as simulation tool and model checker
- Semantics and analysis tool for modeling languages
  - model checker for free for those languages

Needed Future Work

- Combine timed and probabilistic behaviors

Needed Future Work

- Combine timed and probabilistic behaviors
- Scale up model checking; extend sound/complete analysis:
    - symbolic analysis; SMT solving

Needed Future Work

- Combine timed and probabilistic behaviors
- Scale up model checking; extend sound/complete analysis:
    - symbolic analysis; SMT solving
- Counterexamples/witnesses for timed CTL model checking