

# Formalizing and Analyzing Real-Time and Cyber-Physical Systems in Rewriting Logic

---

Peter Csaba Ölveczky

University of Oslo

LAP 2026, Dubrovnik, September 26, 2026

# What is This Talk About?

1. How to model **real-time** systems in rewriting logic
2. Applications

## **Background: Rewriting Logic and Maude**

---



- Logic of concurrent change
- **Data types** defined by **equational specifications**
  - “term rewriting modulo associativity and/or commutativity”
- (Local) change defined by **rewrite rules**

$l : t \longrightarrow t' \text{ if } condition$



- Logic of concurrent change
- Data types defined by equational specifications
  - “term rewriting modulo associativity and/or commutativity”
- (Local) change defined by rewrite rules

$$l : t \longrightarrow t' \text{ if } condition$$

Expressive and general, yet intuitive executable formalism



**Maude**: High-performance tool for rewriting logic

- Explicit-state
  - simulation
  - reachability analysis
  - temporal logic model checking
- Narrowing
- Connections to SMT solvers CVC4 and Yices2
- Meta-programming
- **Statistical model checking** via **umaudemc**
  - **qualitative** and **quantitative** analysis

# Some Applications of Maude

- Logical framework
- **Semantics** for programming and modeling languages
  - C, Java, JVM, Scheme, Solidity, MOF, ...
  - **K framework** (G. Rosu)
- Security
  - found unknown **security attacks** in **Internet Explorer**
  - **Maude-NPA**: Cathy Meadows **NLA Protocol Analyzer**
- **Network protocols** and **cloud computing**
  - Apache Cassandra, Google's Megastore, ZooKeeper, DNS, Mastodon, ...
- **Biological** systems
  - cell biology (Pathway logic)
  - brains (Alzheimer, Parkinson, human cognition, ...)
- ... and much more

# Maude Basics: Object-Based Modeling of Distributed Systems

- Each node in distributed system modeled as an **object**
- State (“configuration”) **multiset** of **objects** and **messages**

## Example: Many Persons

```
omod MANY-PERSONS is protecting NAT . protecting STRING .  
  sort Status .  
  
  op single : -> Status [ctor] .  
  op married : Oid -> Status [ctor] .  
  
  msg divorce : Oid -> Msg .  
  
  class Person | age : Nat, status : Status .  
  
  subsort String < Oid .  
  
  vars X X2 : Oid .  vars N N2 : Nat .  vars S S2 : Status .
```

## Example: Many Persons (cont.)

```
cr1 [birthday] :  
  < X : Person | age : N > => < X : Person | age : N + 1 > if N < 150 .
```

```
cr1 [wedding] :  
  < X : Person | age : N, status : single >  
  < X2 : Person | age : N2, status : single >  
=>  
  < X : Person | status : married(X2) >  
  < X2 : Person | status : married(X) > if N > 15 and N2 > 15 .
```

```
cr1 [death] : < X : Person | > => none .
```

```
rl [wantDivorce] :  
  < X : Person | status : married(X2) >  
=> < X : Person | status : single >  
  divorce(X2) .
```

```
rl [acceptDivorce] :  
  divorce(X)  
  < X : Person | status : married(X2) >  
=> < X : Person | status : single > .
```

## Example: Many Persons (cont.)

A sample state:

```
< "Sue Ellen" : Person | age : 45, status : married("JR") >  
< "Cally" : Person | age : 20, status : single >  
< "JR" :      Person | age : 50, status : single >  
divorce("Sue Ellen")
```

## Example: Many Persons (cont.)

```
Maude> search [1]
      < "Peter" : Person |  age : 25, status : single >
      < "Rachel" : Person | age : 23, status : single >
=>*
      < "Peter" : Person |  age : 70, status : S >
      < "Rachel" : Person | age : 23, status : S2 > .
```

## Example: Many Persons (cont.)

```
Maude> search [1]
      < "Peter" : Person | age : 25, status : single >
      < "Rachel" : Person | age : 23, status : single >
=>*
      < "Peter" : Person | age : 70, status : S >
      < "Rachel" : Person | age : 23, status : S2 > .
```

Solution 1 (state 79906)

S --> single

S2 --> single

# Real-Time Systems in Maude

---

# Modeling Formalisms: Key Trade-off

modeling convenience  $\longleftrightarrow$  analytic power

## Expressive formalism

- + large and complex systems
- properties often undecidable
- limited automatic analysis

## Restricted formalism

- + properties decidable
- + automatic formal analysis
- impossible/hard to model complex systems

## Main challenge

How to deal with dense time?

## Main challenge

How to deal with dense time?

- Model checking for real-time systems
  - timed automata (UPPAAL, RED, ...)
  - time(d) Petri nets
  - timed process algebras
  - BIP
  - ...
- Limited/restricted formalisms (data types? communication? unbounded data structures?)
  - decidability for dense time
  - hard/impossible to capture complex designs

## Real-Time Systems in Rewriting Logic: Remember Problem?

"Peter" could age 50 years, while "Rachel" did not age ...

- “Ordinary” rewrite rules model **instantaneous** change
- “Tick” rewrite rules model **time elapse**:

$$l : \{t\} \longrightarrow \{t'\} \text{ in time } \tau \text{ if } cond$$

- **whole state** has form  $\{u\}$

## Example: Discrete “Clock”

### Example (Discrete clock)

$[reset] : \text{clock}(12) \longrightarrow \text{clock}(0)$

$[tick] : \{\text{clock}(X)\} \longrightarrow \{\text{clock}(X + 1)\} \text{ in time } 1 \text{ if } X < 12$

## Example: What About Dense-time “Clocks”?

How to “cover/capture” dense time domains?

### Example (Dense-time clock)

$[reset] : \text{clock}(12) \longrightarrow \text{clock}(0)$

$[tick] : \{\text{clock}(X)\} \longrightarrow \{\text{clock}(X + T)\} \text{ in time } T$   
if  $T \leq 12 - X$

- $X$  and  $T$  are variables
- not executable (what is  $T$ ?)

- Explixcit-state analysis
- Time sampling of dense-time systems:
  - increase time by fixed amount  $\Delta$  in each tick step
  - increase time maximally in each step

- **Explicit-state** analysis
- **Time sampling** of dense-time systems:
  - increase time by **fixed amount**  $\Delta$  in each tick step
  - increase time **maximally** in each step
- Time-bounded and unbounded
  - reachability analysis
  - LTL model checking
  - **timed CTL** model checking
  - find earliest/latest
- Time-sampling-based analysis not always sound
  - sound and complete for some classes of systems (WRLA'06)

# Real-time Rewrite Theories Directly in Maude

```
sorts System GlobalSystem ClockedSystem .
subsort GlobalSystem < ClockedSystem .

op {_} : System -> GlobalSystem [ctor] .
op _in time_ : GlobalSystem Time -> ClockedSystem [ctor] .
op _in time_ : ClockedSystem Time -> ClockedSystem .

vars T T' : Time .  var STATE : ClockedSystem
eq (STATE in time T) in time T' = STATE in time (T + T') .
```

## Real-time Rewrite Theories in Maude (cont.)

Tick rule “standard” Maude rule

```
cr1 [tick] : {t} => {u} in time  $\Delta$  if cond
```

Tick rule “standard” Maude rule

```
cr1 [tick] : {t} => {u} in time  $\Delta$  if cond
```

- variations for time-bounded and “unclocked” analysis

Tick rule “standard” Maude rule

```
cr1 [tick] : {t} => {u} in time  $\Delta$  if cond
```

- variations for **time-bounded** and “**unclocked**” analysis

### Example

Tick rule corresponding to **maximal** time sampling:

```
cr1 [tick] : {t} => {u} in time T  
if T := mte(t) /\ T > 0 /\ cond
```

Tick rule has the form

```
cr1 [tick] :  
  {STATE} => {timeEffect(STATE, T)} in time T  
  if T := ... /\ T <= mte(STATE) .
```

- **mte**: maximal possible time elapse
- **timeEffect**: what happens when time advances
- define for single objects!

# Object-Oriented Real-Time Systems: Example

```
crl [wedding] : ... => ...
```

```
rl [wantDivorce] :  
  < X : Person | status : married(X2) >  
=>  
  < X : Person | status : single >  
  dly(divorce(X2), 4) .
```

```
rl [acceptDivorce] :  
  divorce(X)  
  < X : Person | status : married(X2) >  
=>  
  < X : Person | status : single > .
```

```
rl [death] : < X : Person | status : S > => none .
```

```
crl [tick] :  
  {STATE} => {timeEffect(STATE, T)} in time T if T := 1 /\ T <= mte(STATE) .
```

## Object-Oriented Real-Time Systems: Example (cont.)

**eq** timeEffect(< X : Person | age : N >, T)  
= < X : Person | age : N + T > .

**eq** mte(< X : Person | age : N >) = 150 minus N .

# Analyzing Persons

```
Maude> search [1, 300]
      {< "Peter" : Person |  age : 25, status : single >
       < "Rachel" : Person | age : 23, status : single >}
=>*
      {< "Peter" : Person |  age : 40, status : S >
       < "Rachel" : Person | age : 23, status : S2 >} in time T .
```

No solution.

# Analyzing Persons

```
Maude> search [1, 300]
      {< "Peter" : Person | age : 25, status : single >
       < "Rachel" : Person | age : 23, status : single >}
=>*
      {< "Peter" : Person | age : 40, status : S >
       < "Rachel" : Person | age : 23, status : S2 >} in time T .
```

No solution.

```
Maude> search [1, 300]
      {< "Peter" : Person | age : 25, status : single >
       < "Rachel" : Person | age : 23, status : single >}
=>*
      {< "Peter" : Person | age : 35, status : S >
       < "Rachel" : Person | age : 33, status : S2 >} in time T .
```

Solution 1 (state 217) S --> single S2 --> single T --> 10

## Analyzing Persons (cont.)

```
Maude> search [1]
      {< "Peter" : Person |  age : 25, status : single >}
=>*
      {< "Peter" : Person |  age : 160, status : S >} in time T .
```

does not terminate (why?)

## Obscure Details

---

Tick rule

`cr1` [*tick*] : {*t*} => {*u*} **in time**  $\Delta$  if *cond*

- states represented as terms {*t*} **in time** *duration*
- infinite reachable “clocked state space” even when finite reachable state space

Tick rule

`cr1 [tick] : {t} => {u} in time  $\Delta$  if cond`

replaced by

`cr1 [tick] : {t} => {u} if cond`

for **unbounded** analysis; and by

`cr1 [tick] : {t} in time T2 => {u} in time (T2 +  $\Delta$ )  
if cond /\ T +  $\Delta$  <= timeBound`

for time-bounded analyses

## State Spaces: Both Unbounded and Time-Bounded Analysis

`cr1 [tick] : {t} => {u} in time  $\Delta$  if cond`

replaced by

`cr1 [tick] : {t} timeBound TI => {u} timeBound (TI -  $\Delta$ )  
if cond /\ TI >=  $\Delta$`

# State Spaces: Both Unbounded and Time-Bounded Analysis

$\text{crl } [\textit{tick}] : \{t\} \Rightarrow \{u\} \text{ in time } \Delta \text{ if } \textit{cond}$

replaced by

$\text{crl } [\textit{tick}] : \{t\} \text{ timeBound TI} \Rightarrow \{u\} \text{ timeBound (TI} - \Delta) \\ \text{if } \textit{cond} \wedge \text{ TI} \geq \Delta$

- **Unbounded** analysis: initial state  $\{\textit{init}\}$  timeBound INF
  - “clocked state space”  $\approx$  “state space”
- **Bounded** analysis: initial state  $\{\textit{init}\}$  timeBound bound

# State Spaces: Both Unbounded and Time-Bounded Analysis

`cr1 [tick] : {t} => {u} in time  $\Delta$  if cond`

replaced by

`cr1 [tick] : {t} timeBound TI => {u} timeBound (TI -  $\Delta$ )  
if cond /\ TI >=  $\Delta$`

- **Unbounded** analysis: initial state `{init}` timeBound INF
  - “clocked state space”  $\approx$  “state space”
- **Bounded** analysis: initial state `{init}` timeBound bound

## Example

```
Maude> search [1]
  {< "Peter" : Person | age : 25, status : single >} timeBound INF
=>*
  {< "Peter" : Person | age : 160, status : S >} timeBound TI .
```

No solution.

# Applications

---

### Uppaal/Timed automata

- **finite** automata with **clocks**
- decidable for dense time
- fixed communication primitive

# Context: Maude vs Uppaal/Timed Automata

## Uppaal/Timed automata

- **finite** automata with **clocks**
- decidable for dense time
- fixed communication primitive

## Maude

- modeling convenience
- 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

# The Main Question

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

## The Main Question

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

Are there systems where Maude's expressiveness needed

and

Maude analysis yields interesting results?

- “Concrete” systems/protocols
- Formal semantics and analysis for modeling languages
- ...

### AER/NCA:

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

### 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:

- 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:

- 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

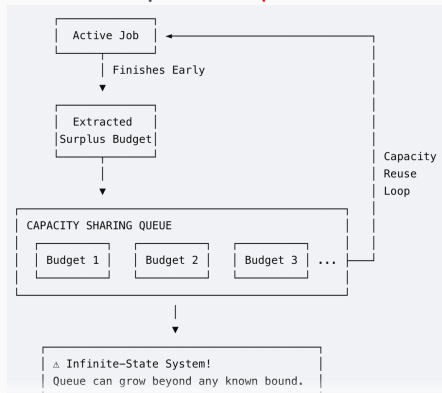
### Key Real-Time Maude features:

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

# Example: CASH Scheduling Algorithm [Ölveczky-Caccamo'06]

**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**: 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**: State-of-the-art scheduling algorithm

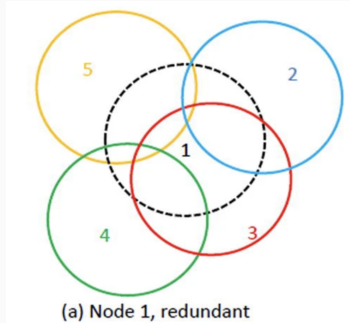
- A job can use more or less time than allocated
- Unused execution times put in a queue for reuse
- Simulation: # elements in queue unbounded
- Search found missed hard deadline
- Extensive “Monte-Carlo simulation” did not find flaw

**Key Real-Time Maude feature:** unbounded data structures

## Example: OGDC Wireless Sensor Network Algorithm [w/ Thorvaldsen]

**OGDC**: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension



## Example: OGDC Wireless Sensor Network Algorithm [w/ Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

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

## Example: OGDC Wireless Sensor Network Algorithm [w/ Thorvaldsen]

OGDC: density control algorithm for wireless sensor networks

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

**OGDC**: density control algorithm for wireless sensor networks

- Simulated by developers using ns-2 with wireless extension
- (Then-)new form of communication: radio transmission
  - easy to specify 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

## Other Applications: Network Protocols

- LMST topology control for wireless sensor networks
- Leader election for MANETS
- EIGRP Cisco routing protocol
- IETF NORM protocol

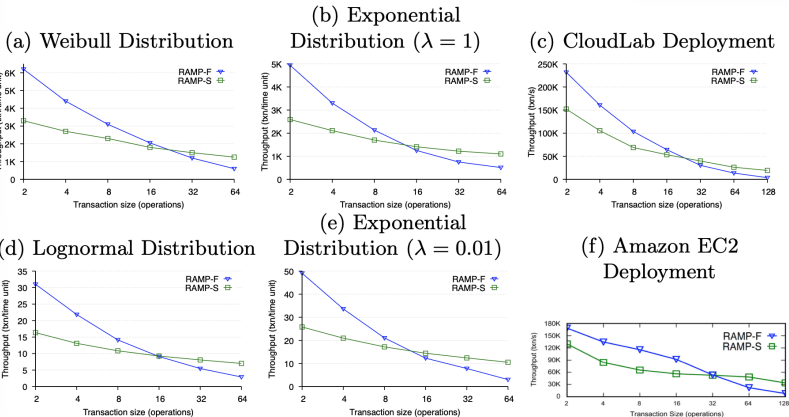
## Other Applications: Cloud-based Transaction Systems

- Google's Megastore
- Apache Cassandra
- Apache ZooKeeper
- RAMP transaction systems
- ROLA (new design)
- P-Store
- ...

# Other Applications: Cloud-based Transaction Systems (cont.)

Key goal: model-based **performance comparison** of design options

- **umaudemc** statistical model checking
- good for **comparing** performance of different designs



## Applications: Transportation Systems

- Parts of AUTOSAR
- European Rail Traffic Management System
- “Leader election” in integrated modular avionics
- Turning control algorithms for airplanes
- **Soft agents** on vehicle platooning
- Find flaw in running cars in Japan

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

# Formal Semantics and Analysis for MDE Languages

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

- **Modeling languages** for embedded systems
  - intuitive domain-specific modeling
  - lack **formal semantics** and **analysis**
- **Maude semantics** and **formal analysis** for such languages
  - modeling languages used in industry
    - **Ptolemy II** DE models
    - AADL avionics modeling standard (subset)
    - UC Berkeley's **Lingua Franca** coordination language for CPSs
    - PLC **Structured Text** language
    - BPMN
    - **(Multirate/Hybrid) Synchronous AADL**
    - DOCOMO's  $\mathcal{L}$  language
  - timed model transformations
    - Real-Time MOMENT-2
    - e-Motions
  - Orc, Timed Rebeca, ...

### Ptolemy II

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

### Ptolemy II

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

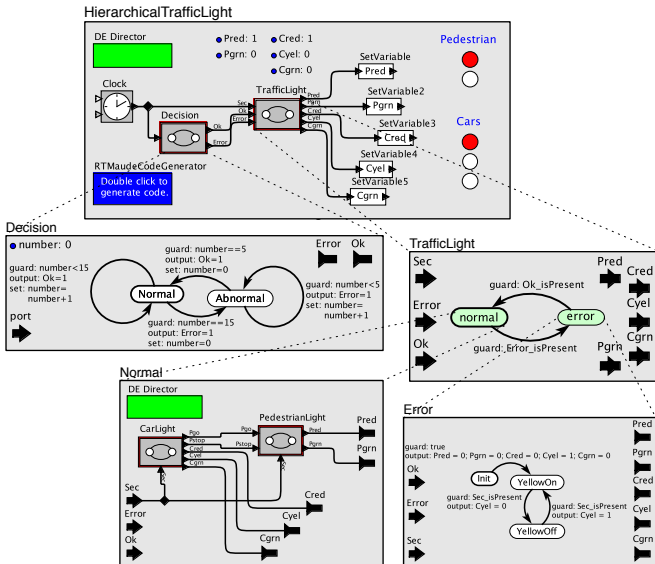
### Ptolemy II

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



# Analyzing Ptolemy II Models Within Ptolemy

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

codeDirectory:

generatorPackage:

generateComment: ☒

inline: ☐

overwriteFiles: ☒

run: ☒

Simulation bound:

Safety Property:

Alternation Property:

Error Handling:

---

Code Generator Commands

Check  
mc-tctl {init} |= AG ((('HierarchicalTrafficLight . 'Decision)|(port 'Error is present)implies AF[<= than 1] 'HierarchicalTrafficLight |('Cyel = # 1,'Cgrn = # 0,'Cred = # 0)) .  
in PTOLEMY-MODELCHECK with mode maximal time increase

Checking equivalent property:  
mc-tctl {init} |= not (E tt U[>= than 0] ('HierarchicalTrafficLight . 'Decision)|(port 'Error is present) and (E not 'HierarchicalTrafficLight |('Cgrn = # 0,'Cred = # 0,'Cyel = # 1)U[>= than 1] tt)) .

Property not satisfied

All Done

# **Case Study: Maude Semantics and Analysis for LINGUA FRANCA**

---

Lingua Franca: Edward A. Lee's coordination language for CPSs



Build **time-sensitive, concurrent, and distributed systems — effortlessly**

Lingua Franca allows you to write blazing-fast, deterministic, multi-threaded and distributed code without any knowledge about threads or synchronization. Focus on your application, not elusive concurrency bugs.

## Lingua Franca: Edward A. Lee's coordination language for CPSs

### Deterministic Concurrency

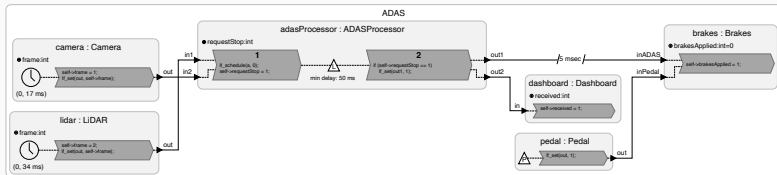
Lingua Franca offers a straightforward way to write multi-threaded applications that ensure determinism by default, eliminating concerns about thread management, synchronization, and race conditions.



### Built-in Timing Semantics

Timing is embedded within Lingua Franca's core, empowering developers to handle time-sensitive tasks with precision and without the complex timing logic typically required in concurrent programming.

# LINGUA FRANCA Example: Automatic Braking Assistance



- **Reaction** code in C, C++, Rust, ...
- LINGUA FRANCA has **logical-time** semantics:
  - reactions executed at **same logical time** happen in **single step**
- **Physical actions** only source of nondeterminism



*Article*

## Verification of Cyberphysical Systems

Marjan Sirjani <sup>1,\*</sup>, Edward A. Lee <sup>2</sup> and Ehsan Khamespanah <sup>3</sup>

<sup>1</sup> School of IDT, Mälardalen University, 722 20 Västerås, Sweden

<sup>2</sup> Department of EECS, University of California at Berkeley, Berkeley, CA 94720-1770, USA; eal@berkeley.edu

<sup>3</sup> Department of ECE, University of Tehran, Tehran 1961733114, Iran; e.khamespanah@ut.ac.ir

\* Correspondence: marjan.sirjani@mdh.se; Tel.: +46-73-662-0517

Received: 31 May 2020; Accepted: 28 June 2020; Published: 2 July 2020



Three programs “hand translated” into extension of Timed Rebeca

## Towards Building Verifiable CPS using Lingua Franca

SHAOKAI LIN, University of California, Berkeley, USA

YATIN A. MANERKAR, University of Michigan, USA

MARTEN LOHSTROH, University of California, Berkeley, USA

ELIZABETH POLGREEN, University of Edinburgh, Scotland

SHENG-JUNG YU, University of California, Berkeley, USA

CHADLIA JERAD, University of Manouba, Tunisia

EDWARD A. LEE and SANJIT A. SESHIA, University of California, Berkeley, USA

(EMSOFT'23)

- $\longrightarrow$  UCLID5  $\longrightarrow$  SMT problem  $\longrightarrow$  Z3
- Bounded safety metric temporal logic model checking

- Event-based (“small-step”) semantics
  - wrong analysis results
  - explores all reaction interleavings in one synchronous step
- No (nondeterministic) physical actions
  - model checking deterministic programs

Edward A. Lee et al.:

*“**formally modeling** the behavior of [discrete-event systems] using a conventional operational model [...] **is difficult** because one needs to specify properties globally over execution traces, a task that cannot be easily achieved at the transition level”*

- Correct “synchronous-reactive” semantics
- Nondeterministic physical actions
- Formal analysis:
  - unbounded and time-bounded:
    - simulation
    - reachability analysis
    - LTL model checking
    - timed CTL model checking (using Real-Time Maude)
  - state and/or dataflow properties

## Benchmarking: LF VERIFIER Benchmark Suite

| Program      | LF Verifier |          | Maude |
|--------------|-------------|----------|-------|
|              | Gen.        | Solving  | Time  |
| ADAS         | 21.07       | 1.72     | 0.008 |
| AircraftDoor | 6.57        | 0.02     | 0.107 |
| Alarm        | 5.36        | 0.01     | 0.030 |
| CoopSchedule | 21.1        | 0.31     | 0.002 |
| Elevator     | 325.14      | 1,038.02 | 0.005 |
| Election     | 41.65       | 3.27     | 0.001 |
| Election2    | 12.21       | 0.17     | 0.001 |
| Factorial    | 21.46       | 13.15    | 0.001 |
| Fibonacci    | 94.96       | 153.49   | 0.010 |
| PingPong     | 13.42       | 0.14     | 0.002 |
| Pipe         | 142.41      | 12.15    | 0.001 |
| ProcessMsg   | 13.23       | 0.74     | 0.001 |
| ProcessSync  | 4.99        | 0.01     | 0.001 |
| Railroad     | 158.14      | 4.33     | 0.003 |
| Ring         | 15.83       | 0.07     | 0.001 |
| RoadsideUnit | 74.07       | 0.62     | 0.001 |
| SafeSend     | 6.48        | 0.03     | 0.001 |
| Subway       | 29.96       | 0.31     | 0.001 |
| Thermostat   | 13.41       | 0.14     | 0.037 |
| TrafficLight | -           | -        | 0.001 |

We have extended the LF compiler to automatically:

- generate Maude model from **annotated LF program**
- perform Maude analysis corresponding to **property annotation**
- present Maude analysis results in **intuitive** form

# Hybrid Systems

---

- Typically modeled as **hybrid automata**
  - very limited control programs
- Symbolic reachability analysis

### HYBRIDSYNCHAADL:

- Model synchronous composition of hybrid components
  - Hybrid PALS → distributed implementation also correct
  - all sampling/actuation times with imprecise local clocks
- Modeling: subset of avionics modeling standard AADL
  - expressive control language
- Continuous dynamics: continuous functions or ODEs
- Modeling and analysis integrated into OSATE

- Formal semantics of HYBRIDSYNCHAADL in Maude+SMT

- Formal semantics of HYBRIDSYNCHAADL in Maude+SMT
- Discrete behaviors encoded in Maude

- Formal semantics of HYBRIDSYNCHAADL in Maude+SMT
- Discrete behaviors encoded in Maude
  - thread behaviors, behavior annex programs, synchronous communication, . . .

- Formal semantics of HYBRIDSYNCHAADL in Maude+SMT
- Discrete behaviors encoded in Maude
  - thread behaviors, behavior annex programs, synchronous communication, . . .
- Continuous behaviors encoded in SMT

- Formal semantics of HYBRIDSYNCHAADL in Maude+SMT
- Discrete behaviors encoded in Maude
  - thread behaviors, behavior annex programs, synchronous communication, ...
- Continuous behaviors encoded in SMT
  - continuous dynamics, sampling/actuation times, clock skews, ...

Maude and Yices2 used to perform formal analysis

- Randomized simulation
  - randomly “samples” clock skews, sampling/actuation times, initial values, ...

Maude and Yices2 used to perform formal analysis

- Randomized simulation
  - randomly “samples” clock skews, sampling/actuation times, initial values, ...
- Symbolic reachability analysis
  - all possible continuous behaviors encoded in SMT

Maude and Yices2 used to perform formal analysis

- Randomized simulation
  - randomly “samples” clock skews, sampling/actuation times, initial values, ...
- Symbolic reachability analysis
  - all possible continuous behaviors encoded in SMT
- Portfolio analysis
  - randomized simulation and symbolic reachability analysis execute concurrently

# Experimental Evaluation

| Model            | Tool    | $Inv_{\top}$ |          |          | $Inv_{\perp}$ |          |          |
|------------------|---------|--------------|----------|----------|---------------|----------|----------|
|                  |         | $N = 2$      | $N = 3$  | $N = 4$  | $N = 2$       | $N = 3$  | $N = 4$  |
|                  |         | Time $B$     | Time $B$ | Time $B$ | Time $B$      | Time $B$ | Time $B$ |
| Rend<br>(single) | HSADL   | 2.0 5        | 3.9 5    | 5.8 5    | 2.4 3         | 4.2 3    | 5.9 3    |
|                  | HyComp  | 0.8 5        | 4.0 5    | 17.2 5   | 8.9 3         | 11.5 3   | 192.6 3  |
|                  | SpaceEx | 8.0 5        | 2230.3 3 | 4.5 1    | 5.1 3         | 2676.6 3 | T/O -    |
|                  | dReach  | 1382.7 3     | 107.1 1  | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 3552.8 4     | 2725.5 2 | 1205.2 1 | 167.3 3       | 380.4 2  | 838.0 3  |
| Form<br>(single) | HSADL   | 3.0 5        | 7.3 5    | 7.9 5    | 15.5 4        | 2.5 2    | 5.2 2    |
|                  | HyComp  | 13.3 5       | 41.3 5   | 182.1 5  | T/O -         | 2.6 2    | 20.3 2   |
|                  | SpaceEx | 91.9 2       | 2.8 1    | 114.8 1  | T/O -         | T/O -    | T/O -    |
|                  | dReach  | 139.0 1      | T/O -    | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 1464.7 2     | 873.4 1  | T/O -    | T/O -         | 45.3 1   | 291.3 2  |
| Thermostat       | HSADL   | 2.7 5        | 4.7 5    | 7.8 5    | 7.6 5         | 15.3 5   | 10.7 4   |
|                  | HyComp  | 1.6 5        | 8.5 5    | 37.9 5   | 2.6 5         | 15.5 5   | 43.1 4   |
|                  | SpaceEx | 2.3 5        | 696.4 3  | 34.5 1   | 2.2 5         | T/O -    | T/O -    |
|                  | dReach  | 341.6 3      | 57.5 1   | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 3196.4 5     | 1240.7 2 | 977.7 1  | 15.5 3        | 1718.1 4 | T/O -    |
| Rend<br>(double) | HSADL   | 3.7 4        | 37.8 4   | 6.9 4    | 1.4 2         | 16.3 2   | 2.8 2    |
|                  | SpaceEx | 1147.6 3     | 81.1 1   | T/O -    | 15.2 2        | T/O -    | T/O -    |
|                  | dReach  | 2156.2 3     | 274.3 1  | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 232.5 2      | 230.1 1  | T/O -    | 2.2 2         | 25.4 2   | 2613.8 1 |

Timeout: 3,600  
seconds

$N$ : # components

$B$ : # iterations

$Inv_{\top}$ : largest  $B$  for  
which tool  
could analyze

$Inv_{\perp}$ : smallest  $B$   
where  
counterexample  
found

# Experimental Evaluation

| Model            | Tool    | $Inv_{\top}$ |          |          | $Inv_{\perp}$ |          |          |
|------------------|---------|--------------|----------|----------|---------------|----------|----------|
|                  |         | $N = 2$      | $N = 3$  | $N = 4$  | $N = 2$       | $N = 3$  | $N = 4$  |
|                  |         | Time $B$     | Time $B$ | Time $B$ | Time $B$      | Time $B$ | Time $B$ |
| Rend<br>(single) | HSADDL  | 2.0 5        | 3.9 5    | 5.8 5    | 2.4 3         | 4.2 3    | 5.9 3    |
|                  | HyComp  | 0.8 5        | 4.0 5    | 17.2 5   | 8.9 3         | 11.5 3   | 192.6 3  |
|                  | SpaceEx | 8.0 5        | 2230.3 3 | 4.5 1    | 5.1 3         | 2676.6 3 | T/O -    |
|                  | dReach  | 1382.7 3     | 107.1 1  | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 3552.8 4     | 2725.5 2 | 1205.2 1 | 167.3 3       | 380.4 2  | 838.0 3  |
| Form<br>(single) | HSAADL  | 1            |          |          |               |          |          |
|                  | HyComp  | 9            |          |          |               |          |          |
|                  | SpaceEx | 13           |          |          |               |          |          |
|                  | dReach  | 146          |          |          |               |          |          |
|                  | Flow*   |              |          |          |               |          |          |
| Thermostat       | HSAADL  |              |          |          |               |          |          |
|                  | HyComp  |              |          |          |               |          |          |
|                  | SpaceEx |              |          |          |               |          |          |
|                  | dReach  | 341.6 3      | 57.5 1   | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 3196.4 5     | 1240.7 2 | 977.7 1  | 15.5 3        | 1718.1 4 | T/O -    |
| Rend<br>(double) | HSAADL  | 3.7 4        | 37.8 4   | 6.9 4    | 1.4 2         | 16.3 2   | 2.8 2    |
|                  | SpaceEx | 1147.6 3     | 81.1 1   | T/O -    | 15.2 2        | T/O -    | T/O -    |
|                  | dReach  | 2156.2 3     | 274.3 1  | T/O -    | T/O -         | T/O -    | T/O -    |
|                  | Flow*   | 232.5 2      | 230.1 1  | T/O -    | 2.2 2         | 25.4 2   | 2613.8 1 |

Timeout: 3,600  
seconds

$N$ : # components

$B$ : # iterations

## From Experimental Results

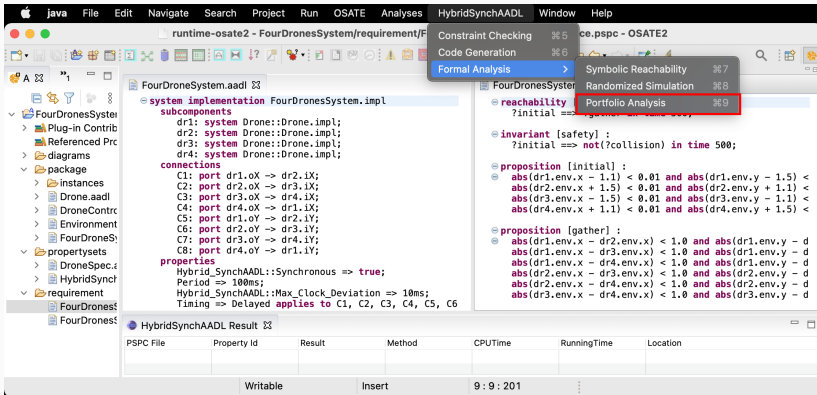
HybridSynchAADL is effective for analyzing models with **both** complex control programs and continuous behaviors

counterexample  
found

## The HybridSynchAADL Tool: Example

The screenshot shows the HybridSynchAADL IDE interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, OSATE, Analyses, HybridSynchAADL, Window, and Help. The title bar indicates the project is 'runtime-osate2 - FourDronesSystem/package/FourDroneSystem.aadl - OSATE2'. The left sidebar shows the project structure, with 'FourDroneSystem' selected. The main editor displays the code for 'FourDroneSystem.impl', which defines four drone components (dr1, dr2, dr3, dr4) and their interconnections. The right pane shows the corresponding block diagram for 'FourDronesSystem.impl', illustrating the data flow between the drone components. The bottom pane shows the 'HybridSynchAADL Result' table, which is currently empty.

## The HybridSynchAADL Tool: Example



# The HybridSynchAADL Tool: Example

The screenshot displays the HybridSynchAADL tool interface. The top menu bar includes options like java, File, Edit, Navigate, Search, Project, Run, OSATE, Analyses, HybridSynchAADL, Window, and Help. The main workspace is divided into three panes:

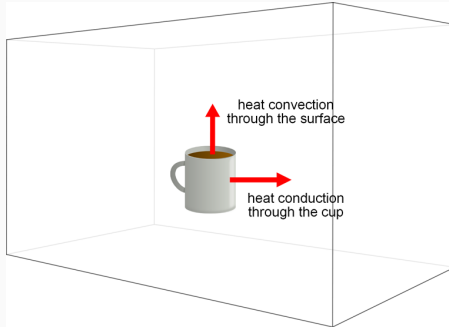
- Left Pane:** A project tree showing the structure of the 'FourDronesSystem' package, including subcomponents, instances, and properties.
- Center Pane:** The 'FourDronesSystem.impl' file, showing the implementation of the system. It includes subcomponents (dr1, dr2, dr3, dr4), connections (C1 through C8), and properties (Hybrid\_SynchAADL::Synchronous, Period, Max\_Clock\_Deviation, Timing).
- Right Pane:** The 'FourDronesSystem\_impl\_Instance.pspc' file, showing the verification results for the system. It includes properties like 'reachability [rendezvous]', 'invariant [safety]', 'proposition [initial]', and 'proposition [gather]'.

At the bottom, a table displays the results of the verification run, with a red border highlighting the first two rows:

| PSPC File                           | Property Id | Result               | Method   | CPUTime | RunningTime | Location                                    |
|-------------------------------------|-------------|----------------------|----------|---------|-------------|---------------------------------------------|
| FourDronesSystem_impl_Instance.pspc | rendezvous  | Reachable            | symbolic | 1872ms  | 1792ms      | <a href="#">FourDronesExample/verificat</a> |
| FourDronesSystem_impl_Instance.pspc | safety      | Counterexample found | random   | 1502ms  | 1673ms      | <a href="#">FourDronesExample/verificat</a> |

Complex **hybrid** systems:

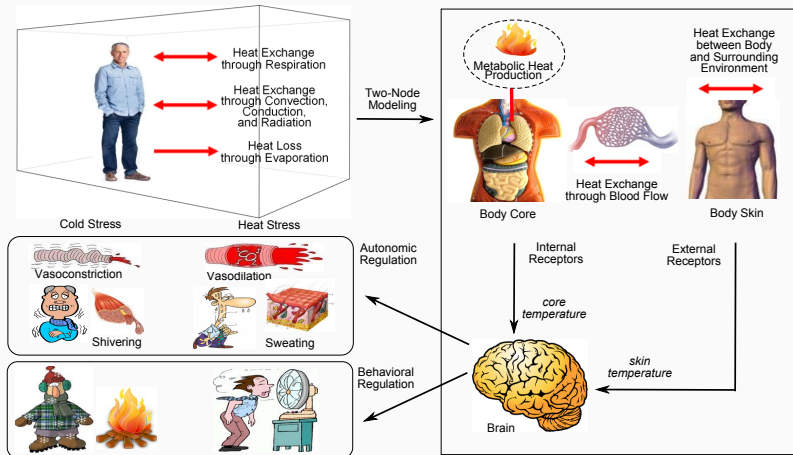
**Multiple** physical components **interact** and influence each other's continuous behaviors



# Case Study II: HI-Maude [Fadlisyah-Ölveczky-Ábrahám'14]

Complex **hybrid** systems:

**Multiple** physical components **interact** and influence each other's **continuous behaviors**



## HI-Maude (II)

- Nightmare to specify continuous dynamics
- Not object-oriented
  - define continuous dynamics for each configuration

## HI-Maude (II)

- Nightmare to specify continuous dynamics
- Not object-oriented
  - define continuous dynamics for each configuration

### HI-Maude

Physical components and physical interactions modeled as objects

- define continuous behavior on single components and interactions
- object-oriented! (continuous behaviors defined at class level)
- dynamic creation/deletion of objects
- numerical methods approximate continuous behaviors

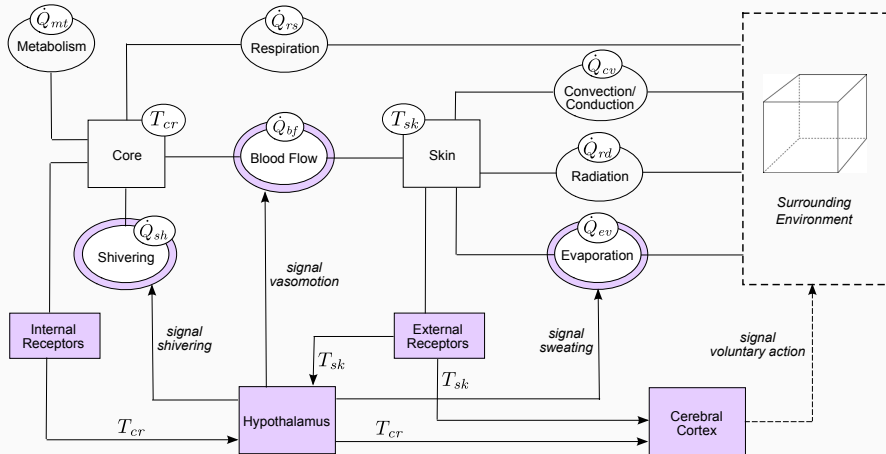
# HI-Maude Case Study: What Happened at the 2010 Sauna World Championships?



# HI-Maude Case Study: What Happened at the 2010 Sauna World Championships?

- Model human thermoregulation
- Analyze how long a human can endure in sauna

# HI-Maude Case Study: What Happened at the 2010 Sauna World Championships?



# Symbolic Analysis

---

Dense time:

$tick : \{t\} \longrightarrow \{t'\}$  in time  $T$  if  $cond$

- Instead of **sampling** concrete values of  $T$  ...
- ... represent time increase **symbolically**
- **new** SMT variable for each tick rule application

- Maude+SMT semantics and analysis for
  - parametric timed automata
  - parametric interval time Petri nets

- Maude+SMT semantics and analysis for
  - parametric timed automata
  - parametric interval time Petri nets
- With **folding** and **quantifier elimination**:
  - Maude+SMT reachability analysis terminates whenever Imitator/Roméo analyses terminate!

## **Take-Home Message**

---

## Take-Home Message

- Sophisticated real-time systems easily modeled in Maude
  - expressive, intuitive, executable formalism
  - systems beyond other real-time model checkers

# Take-Home Message

- Sophisticated real-time systems easily modeled in Maude
  - expressive, intuitive, executable formalism
  - systems beyond other real-time model checkers
- Used on **state-of-the-art** systems in different domains
  - value added to domain-specific analysis
  - **SMC** with **umaudemc** give good early model-based **performance prediction** of industrial designs

# Take-Home Message

- Sophisticated real-time systems easily modeled in Maude
  - expressive, intuitive, executable formalism
  - systems beyond other real-time model checkers
- Used on **state-of-the-art** systems in different domains
  - value added to domain-specific analysis
  - **SMC** with **umaudemc** give good early model-based **performance prediction** of industrial designs
- **Semantics** and **analysis tool** for **modeling languages**
  - model checker for free for those languages

# Take-Home Message

- Sophisticated real-time systems easily modeled in Maude
  - expressive, intuitive, executable formalism
  - systems beyond other real-time model checkers
- Used on **state-of-the-art** systems in different domains
  - value added to domain-specific analysis
  - **SMC** with **umaudemc** give good early model-based **performance prediction** of industrial designs
- **Semantics** and **analysis tool** for **modeling languages**
  - model checker for free for those languages
- **Symbolic** analysis with Maude-with-SMT
  - sound and complete analysis for PTAs and PITPNs