The Timed *P* Transformation for Distributed Real-Time Systems

José Meseguer[1] and **Peter Ölveczky**[2]

[1]University of Illinois at Urbana-Champaign

[2]University of Oslo, Norway

WRLA26, April 11, 2026

(Work performed as part of the Maude-HCS project)

The Timed P Transformation

Timed P transformation?

Preliminaries and Motivation

Preliminaries: The P Transformation

Bridging the Semantic Gap between Qualitative and Quantitative Models of Distributed Systems

SI LIU, ETH Zürich, Switzerland
JOSÉ MESEGUER, University of Illinois Urbana-Champaign, USA
PETER CSABA ÖLVECZKY, University of Oslo, Norway
MIN ZHANG, East China Normal University, China
DAVID BASIN, ETH Zürich, Switzerland

(OOPSLA'22)

P: Motivation

Large distributed systems: correctness and performance critical

Example (Google's Megastore)

Correctness: consistency over entity groups

Performance:
- transaction latency
- % of committed transactions

P: Motivation

Large distributed systems: correctness and performance critical

Example (Google's Megastore)

Correctness: consistency over entity groups

Performance:
- transaction latency
- % of committed transactions

Example (Apache Cassandra)

Correctness: eventual consistency

Performance: % of transactions satisfying strong consistency

P: Motivation

Large distributed systems: correctness and performance critical

Example (Google's Megastore)

Correctness: consistency over entity groups

Performance:
- transaction latency
- % of committed transactions

Example (Apache Cassandra)

Correctness: eventual consistency

Performance: % of transactions satisfying strong consistency

Example (Amazon Web Services (SSS, DynamoDB))

Correctness: no user data lost

Performance:
- 99.99% data availability
- transaction latency

P: Motivation

Large distributed systems: correctness and performance critical

Example (Apache Cassandra)

Correctness: eventual consistency

Performance: % of transactions satisfying strong consistency

Example (Amazon Web Services (SSS, DynamoDB))

Correctness: no user data lost

Performance:
- 99.99% data availability
- transaction latency

Example (Facebook)

Correctness:
- causal consistency of posts/comments
- read atomicity of friendships

Performance: important

Challenge: Analyzing Correctness and Performance

Early estimation of correctness and performance:

Correctness: often untimed and non-probabilistic models

- TLA+, Spin, BIP?, Maude, separation logic, Hoare logic, ...

Performance: often timed and/or probabilistic

- finite automata with time/probabilities
- time(d) Petri nets
- network simulation tools (ns-2, ...)
- deployment on clusters

Disadvantages

1. Two very different models/artifacts must be developed
2. Relation between two models/artifacts?
3. Can a team do both?

Example (Amazon Web Services)

TLA+ models could not reason about performance

Rewriting Logic + Maude + (P)VeStA to the Rescue

- Rewriting logic/Maude can analyze correctness of sophisticated distributed systems
  - RAMP transactions, Apache Cassandra, ...

Rewriting Logic + Maude + (P)VeStA to the Rescue

- Rewriting logic/Maude can analyze correctness of sophisticated distributed systems
  - RAMP transactions, Apache Cassandra, ...
- "Performance models:" probabilistic [real-time] rewrite theories

Rewriting Logic + Maude + (P)VeStA to the Rescue

- Rewriting logic/Maude can analyze correctness of sophisticated distributed systems
  - RAMP transactions, Apache Cassandra, ...
- "Performance models:" probabilistic [real-time] rewrite theories
- (P)VeStA analyze such models by statistical model checking

Statistical Model Checking (SMC)

- "Verify" properties up to some confidence level
- PCTL: inefficient (VeStA only?)
- QuaTEx: estimate expected value of computation
  - more efficient
  - run many simulations (possibly in parallel)
  - VeStA, MultiVeStA, PVeStA, umaudemc, ...

Statistical Model Checking (SMC)

- "Verify" properties up to some confidence level
- PCTL: inefficient (VeStA only?)
- QuaTEx: estimate expected value of computation
  - more efficient
  - run many simulations (possibly in parallel)
  - VeStA, MultiVeStA, PVeStA, umaudemc, ...
- SMC requires purely probabilistic models
  - no unquantified nondeterminism

Early Idea: Actor PMaude

PMaude: Rewrite-based Specification Language for Probabilistic Object Systems

Gul Agha[1] José Meseguer[2] Koushik Sen[3]

Department of Computer Science,
University of Illinois at Urbana Champaign, USA.

- Untimed correctness model
- Receive message $\longrightarrow$ send messages
- Probabilistic timed model: msg delay sampled from continuous distribution $\longrightarrow$ almost-surely no nondeterminism

Early Idea: Actor PMaude

PMaude: Rewrite-based Specification Language for Probabilistic Object Systems

Gul Agha[1] José Meseguer[2] Koushik Sen[3]

Department of Computer Science,
University of Illinois at Urbana Champaign, USA.

- Untimed correctness model
- Receive message $\longrightarrow$ send messages
- Probabilistic timed model: msg delay sampled from continuous distribution $\longrightarrow$ almost-surely no nondeterminism

- does not relate correctness model with performance model
- non-message-triggered rules sometimes needed

Actor PMaude Ideas Take Off

- "Probabilistic simulation model" developed by hand from untimed nondeterministic model
  - error-prone
  - ugly (fake messages, etc.)
  - purely probabilistic?
- PVeStA performance estimates good for comparing system designs

The P Transformation

Bridging the Semantic Gap between Qualitative and Quantitative Models of Distributed Systems

SI LIU, ETH Zürich, Switzerland
JOSÉ MESEGUER, University of Illinois Urbana-Champaign, USA
PETER CSABA ÖLVECZKY, University of Oslo, Norway
MIN ZHANG, East China Normal University, China
DAVID BASIN, ETH Zürich, Switzerland

- "Actor PMaude" and "object-triggered" rules
- Formalized source models: generalized actor rewrite theories

The P Transformation

Bridging the Semantic Gap between Qualitative and Quantitative Models of Distributed Systems

SI LIU, ETH Zürich, Switzerland
JOSÉ MESEGUER, University of Illinois Urbana-Champaign, USA
PETER CSABA ÖLVECZKY, University of Oslo, Norway
MIN ZHANG, East China Normal University, China
DAVID BASIN, ETH Zürich, Switzerland

- "Actor PMaude" and "object-triggered" rules
- Formalized source models: generalized actor rewrite theories
- P transformation $(\mathcal{R}, \Pi) \mapsto \mathcal{R}_\Pi$
  - $\mathcal{R}$ generalized actor rewrite theory
  - Π parametric continuous prob. distribution for message delays
  - $\mathcal{R}_\Pi$ probabilistic timed rewrite theory

The P Transformation

Bridging the Semantic Gap between Qualitative and Quantitative Models of Distributed Systems

SI LIU, ETH Zürich, Switzerland
JOSÉ MESEGUER, University of Illinois Urbana-Champaign, USA
PETER CSABA ÖLVECZKY, University of Oslo, Norway
MIN ZHANG, East China Normal University, China
DAVID BASIN, ETH Zürich, Switzerland

- "Actor PMaude" and "object-triggered" rules
- Formalized source models: generalized actor rewrite theories
- P transformation $(\mathcal{R}, \Pi) \mapsto \mathcal{R}_\Pi$
  - $\mathcal{R}$ generalized actor rewrite theory
  - Π parametric continuous prob. distribution for message delays
  - $\mathcal{R}_\Pi$ probabilistic timed rewrite theory
- Key theorems proved:
  1. $\mathcal{R}_\Pi$ + *init* almost-surely purely probabilistic
  2. All simulations of $\mathcal{R}_\Pi$ are also behaviors of $\mathcal{R}$

More Transformations

- $\mathcal{R}_\Pi \mapsto \mathit{Sim}(\mathcal{R}_\Pi)$ gives executable rewrite theory
- M transformation: observe events

More OOPSLA'22: Actors2PMaude Tool

Actors2PMaude tool:

- implement transformations
- run PVeStA
- good predictive power for larger distributed systems
- in Python
- brittle (?)

The Timed P Transformation

The Timed P Transformation

- P transformation starting point:
  - untimed nondeterministic model
  - deterministic rewrite rules
  - no message loss

The Timed *P* Transformation

- *P* transformation starting point:
  - untimed nondeterministic model
  - determinististic rewrite rules
  - no message loss
- Sometimes start with timed model (why?)
  - most IETF protocols use timers
    - don't always want to transform to untimed model
  - extend timed model (e.g., DNS in Maude-HCS project)
  - untimed model too general?
  - correctness depends on time values
  - evaluate for different periods/time values

The Timed P Transformation: Input Models

Generalized actor real-time rewrite theories (?)

- Timed system: timers and clocks
- Object-based
- Message-triggered and timer-triggered rules
- "Standard" tick rule
- Nondeterministic message loss
- Cyber-physical systems: sample values nondeterministically
- Finitary nondeterminism in rules
- Nondeterministic message delay ($\in [0, \infty)$)

Generalized Timed Actor Real-time Rewrite Theories

Deterministic message-triggered and timer-triggered rules

```
[l] : (to o [from o'] : mp) < o : Cl | atts >
  =>                          < o : Cl | atts' >  [msgs]  [if cond].
```

```
[l] : < o : Cl | timers : [0 ; p ; td] otherTimers, atts >
  => < o : Cl | timers : [v ; p' ; td] otherTimers', atts' > [msgs] [if cond].
```

```
sorts MsgList STask .  subsort Msg < MsgList .
subsort STask < Configuration .
op nil : -> MsgList [ctor] .
op _;_ : MsgList MsgList -> MsgList [ctor assoc id: nil] .
op [_] : MsgList -> STask [ctor] .
```

Network Transmission Rules (with Message Loss)

```
[sendDelayed] : [M ; ML] => dly(M, T)   [ML]   [nonexec] .
[sendNil] :      [nil] => none .

[msgArrival] :  dly(M,0)  => if B then M else none fi [nonexec] .
```

Timer-triggered Sensor-reading rule

```
[read-sensor-I] :
   < O : Plant | timers : [0 ; p ; td] TIMERS, values : V, atts >
=> < O : Plant | timers : [p ; p ; td] TIMERS, values : NEW-V, atts' >
   [msgs]     [nonexec] .
```

Non-executable Models

Unlike P translation, source models not (directly) executable

Requirements

Overall Goal

Add probability distributions for nondeterministic features (new variables), to obtain almost-surely purely probabilistic systems

- At most one timer (re)set in any rule; others may be turned off
- Timers can only be set to positive natural numbers
- At most one timer is active in *init* and ...
- Unique rule (+unique substitution of LHS variables) enabled when message arrives or timer expires
- ...

The Timed P Transformation: Inputs (II)

Input: Generalized actor real-time rewrite theories satisfying requirements + probability distributions Π:

- for each rule l sending messages: parametric continuous distribution(s) $\pi_l(\vec{x})$ (for message delays)
- $\pi_{messageArrival_l}(\vec{x})$ on the Booleans
- $\pi_{readSensor_l}(\vec{x})$ for the new sensor values
- distributions on the finitary nondeterminism
- continuous distribution π_{init} on the delays of messages in $init$

The Timed P Transformation: Outputs

Output: probabilistic rewrite theory $\mathcal{R}_\Pi$ and initial state $init_\Pi$

- instantiate new variables according to probability distributions

$$[l] : (\texttt{to}\ o\ [\texttt{from}\ o']\ :\ mp)\ \texttt{<}\ o\ :\ Cl\ \texttt{|}\ atts\ \texttt{>}$$
$$\texttt{=>}\qquad\qquad \texttt{<}\ o\ :\ Cl\ \texttt{|}\ atts'\ \texttt{>}\ \ \texttt{msgDly}_l(\vec{x}, [msgs])\ [\texttt{if}\ cond]$$

$$[\texttt{sendDelayed}_l]\ :$$
$$\texttt{msgDly}_l(\vec{x}, [\texttt{M}\ ;\ \texttt{ML}])\ \texttt{=>}\ \texttt{dly(M,T)}\ \ \texttt{msgDly}_l(\vec{x}, [\texttt{ML}])$$
$$\textit{with probability}\ \texttt{T} := \pi_l(\vec{x})\ .$$

$$[\texttt{msgArrival}]\ :$$
$$\texttt{dly(M,0)}\ \texttt{=>}\ \texttt{if B then M else none fi}\ \textit{with probability}\ \texttt{B} := \pi_{\texttt{msgArrival}}(\texttt{M})$$

$$[\texttt{read-sensor-}l]\ :$$
$$\texttt{<}\ o\ :\ Plant\ \texttt{|}\ \texttt{values : V,}\quad \texttt{timers : [0 ; P ;}\ td\texttt{]}\ \texttt{TIMERS,}\ atts\ \texttt{>}$$
$$\texttt{=>}\ \texttt{<}\ o\ :\ Plant\ \texttt{|}\ \texttt{values : NEW-V, timers : [P ; P ;}\ td\texttt{]}\ \texttt{TIMERS,}\ atts'\ \texttt{>}$$
$$\texttt{msgDly}_{readSensors_l}(\vec{x},\ [msgs])\ [\texttt{if}\ cond]$$
$$\textit{with probability}\ \texttt{NEW-V} := \pi_{readSensors_{l,val}}(\vec{x})\ .$$

Key Theorems Proved

1. The "behaviors" of $\mathcal{R}_\Pi$ are related to those in $\mathcal{R}$
2. $\mathcal{R}_\Pi$ (with initial state $init_\Pi$) is almost-surely purely probabilistic (absence of nondeterminism)

Formal Methodology

1. Design capture
2. Execution and testing
   - can choose from finite set of values for new variables
3. Symbolic execution and reachability analysis using Maude-with-SMT
4. "Probabilistic" simulation of $Sim(\mathcal{R}_\Pi)$
5. Statistical model checking of $Sim(\mathcal{R}_\Pi)$
   - or $\mathcal{R}_\Pi$ with QMaude strategies

Example: Round Trip Time Protocol in QMaude

Periodically finding the round trip time between nodes

Example: Round Trip Time Protocol in QMaude

Periodically finding the round trip time between nodes

Define "observers":

```
omod PROPERTIES is extending PROB-SINGLE-RTT .
  including CONVERSION .
  op done : ClockedSystem -> Bool [frozen] .
  var REST : Configuration .  var S : Oid .
  vars T1 T2 : Time .
  eq done({REST < S : Sender | rtt : T1 >} in time T2) = true .
  eq done({REST < S : Sender | rtt : ∞ >} in time T2) = false .

  op rttValue : ClockedSystem -> Float [frozen] .
  eq rttValue({REST < S : Sender | rtt : T1 >} in time T2) = float(T1) .
  eq rttValue({REST < S : Sender | rtt : ∞ >} in time T2) = 0.0 .
endom
```

Example: Round Trip Time Protocol in QMaude

QMaude strategy to sample the values of new variable T-NEW

```
smod RTT-STRAT is extending PROPERTIES .
  strat minStrat @ ClockedSystem .
  vars F F1 F2 : Float .    var S : ClockedSystem .  var T-NEW : Time .
  sd myStrat :=
   ((sample F2 := norm(35.0,0.5) in sendDelayed-send[T-NEW <- rat(F2)]) |
    (sample F2 := norm(11.0,1.0) in sendDelayed-respond[T-NEW <- rat(F2)]) |
    all) .
endsm
```

Example: Round Trip Time Protocol in QMaude: QuaTEx

Find first recorded RTT value:

```
Result() = if s.rval("done(S)")
           then s.rval("rttValue(S)") else  #Result() fi ;

eval E[Result()];
```

Example: Round Trip Time Protocol in QMaude: SMC

```
$ umaudemc scheck pi-transformed-rtt.maude init1 rtt.quatex myStrat
          --assign step

Number of simulations = 30
 μ = 45.90292632697745     σ = 0.9886934410590363     r = 0.369184198263
```

```
$ umaudemc scheck pi-transformed-rtt.maude init1 rtt.quatex myStrat
          -a 0.1 -d 0.1 --assign step

Number of simulations = 330
 μ = 46.04172219843193     σ = 1.073419422904671     r = 0.097468515082
```

Concluding Remarks

Concluding Remarks

- Timed P transformation:
  - from nondeterministic (non-executable) timed "verification model" ...
  - to timed probabilistic "performance estimation model"
- More nondeterminism than P transformation
  - sensor values
  - message loss
  - finitary nondeterminism

Concluding Remarks

- Timed P transformation:
  - from nondeterministic (non-executable) timed "verification model" ...
  - to timed probabilistic "performance estimation model"
- More nondeterminism than P transformation
  - sensor values
  - message loss
  - finitary nondeterminism
- Proved
  - relation between behaviors in $\mathcal{R}$ and $\mathcal{R}_\Pi$
  - absence of nondeterminism in $\mathcal{R}_\Pi$

Immediate Future Work

- Implement Timed P transformation (and Sim and M)
  - integrate into `umaudemc`
- Study expressiveness and convenience of input models