Quantitative Modeling in Maude

Carolyn Talcott
SRI International
MQLA 2009

PLAN

- RWL and Maude
- Reasoning about Time: RealTime Maude
- Probablistic modeling: PMaude
- XTune

What is Rewriting Logic

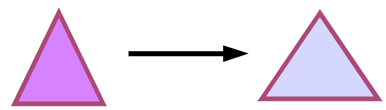
- A logic for executable specification and analysis of concurrent, distributed and/or mobile systems
- A logic to specify other logics or languages
- An extension of equational logic with local rewrite rules expressing
  - concurrent change over time
  - inference rules

Rewrite Theories

- Rewrite theory: (Signature, RewriteRules)
- Signature: (Sorts, Ops, Equations) -- an equational theory describing system state
- Rewrite rule: *label: t => t' if cond*
- Rewriting operates modulo equations
- Generates computations / deductions

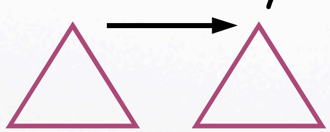
Deduction Rules

one step rewrite:

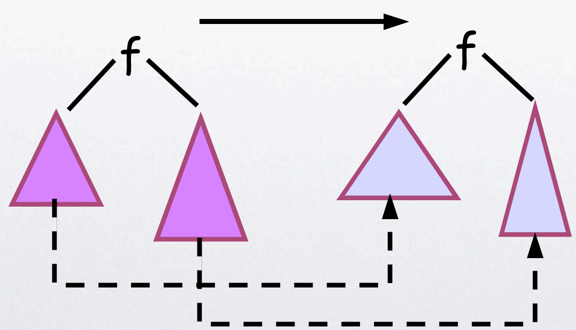


closed under

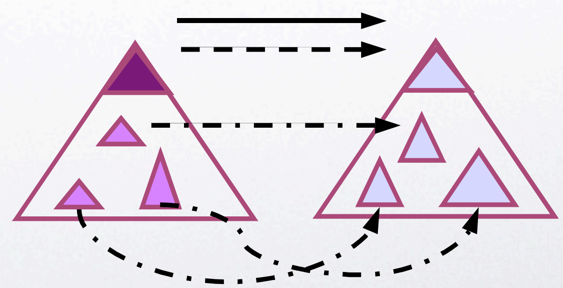
reflexivity:



congruence:



replacement:

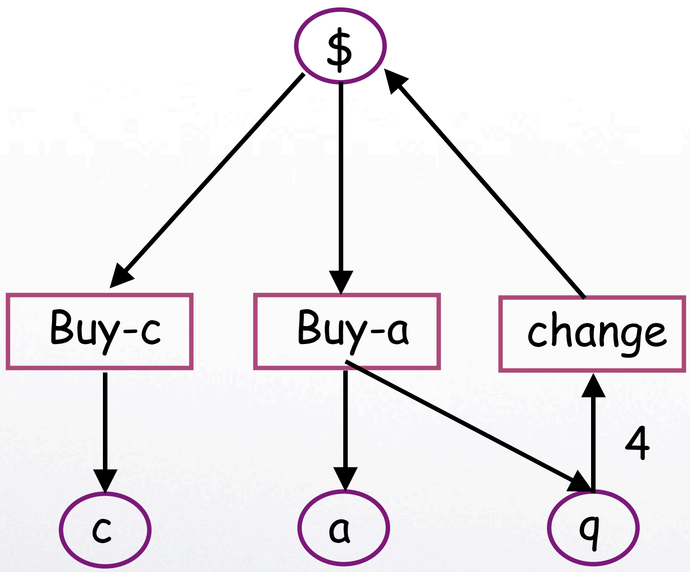


Maude

http://maude.cs.uiuc.edu

- Maude is a language and tool based on RWL
  - High performance rewriting modulo axioms
  - Modularity, builtins, reflection
  - Execution, search, model checking

Petri Net Model of a Vending Machine




```
mod VENDING-MACHINE is
  sorts Coin Item Place Marking .
  subsorts Coin Item < Place < Marking .
  op null : -> Marking .
            *** empty marking
  ops $ q : -> Coin .
  ops a c : -> Item .
  op _ _ : Marking Marking -> Marking
           [assoc comm id: null] .
           *** multiset
  rl[buy-c]: $ => c .
  rl[buy-a]: $ => a q .
  rl[change]: q q q q => $ .
endm
```

Execution and search

What is one way to use 3 $s?

```
Maude> rew $ $ $ .
result Marking: q a c c
```

How can I get 2 apples with 3 $s?

```
Maude> search $ $ $ =>! a a M:Marking .

Solution 1 (state 8)
M:Marking --> q q c

Solution 2 (state 9)
M:Marking --> q q q a

No more solutions.
states: 10  rewrites: 12)
```

Model checking

Starting with 5 $s, can we get 6 apples without accumulating more than 4 quarters?

```
eq vm(M) |= nApples(n) = countPlace(M,a) == n .
eq vm(M) |= lte4Q = countPlace(M,q) <= 4 .
```

```
Maude> red modelCheck(vm($ $ $ $ $),
                  []~(lte4Q U nApples(6)) .
result ModelCheckResult: counterexample(...)
```

Is value conserved?

```
Maude> red modelCheck(vm($ $ $ $ $),[]val(20) .
result Bool: true
```

Real Time Rewrite Theories & Real-Time Maude

Real Time Rewrite Theory (RTRwT)

- RT = ((S,O), E, R, φ, τ)
  - (((S,O),E),R) is an ordinary Rewrite Theory
  - φ interprets a abstract notion of time
  - τ maps rules to terms of sort Time
    - τ(l) > 0 -- a tick rule,
    - τ(l) = 0 -- instantaneous rule
  - R -- l: t => t' in time τ(l) if cond
- Computations/derivations: RT |= t -r-> t'
  - each step instantiates rule, picks a time
  - r is the sum of the times of individual steps

Clock example

R, R’ range over Time, $\tau_{running}$ = R' ...

crl[running]:

{clock(R)} => {clock(R + R')} in time R' if R' <= 24 monus R

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

rl[batterydies]: {clock(24)} => {stopped-clock(24)}

rl[stopped]:

{stopped-clock(R)} => {stopped-clock(R + R')} in time R'

Analysis

- Property logic: rtLTL
  - propositional LTL without Next
  - propositions may refer to time
- Analyses [possibly time bounded]
  - execution
  - search
  - model checking

Sampling

To execute, a strategy is needed to pick times

- Transform RT to $RT^{maxDef(r)}$ (mte sampling)
  - time picked is max allowed by rule condition
  - r is used for the max for unbounded rules

Completeness for mte sampling

$$RT, t_0 \models \Phi \quad \text{iff} \quad RT^{maxDef(r)}, t_0 \models \Phi$$

if RT is time-robust, atoms of Φ are tick-invariant

RealTime Maude

http://www.ifi.uio.no/RealTimeMaude

tick rule form: conf => delta(conf,R’) in time R’ if R’ ≤ mte(conf)

Clock ticks:

crl[running]: {clock(R)} => {clock(R + R')} in time R' if R' <= 24 monus R

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

For running and stopped: delta({clock(R)},R’) = {clock(R + R')}
For running: mte({clock(R)} = 24 monus R
For stopped: mte({clock(R)} = INF

There are simple conditions on delta and mte that guarantee time-robustness
Frequently properties are tick-invariant because they don’t mention variables/attributes changed by delta.

Clock analyses

```
(tsearch [1] {clock(0)} =>* {clock(X:Time)}
               such that X:Time > 24 in time <= 99 .)

eq {stopped-clock(R)} |= clock-dead = true .
eq {clock(R)} |= clock-is(R') = (R == R') .
eq {clock(R)} in time R' |= clockEqualsTime = (R == R') .

(mc {clock(0)} |=t clockEqualsTime U
           (clock-is(24) \/ clock-dead) in time <= 1000 .)
```

Example analyses

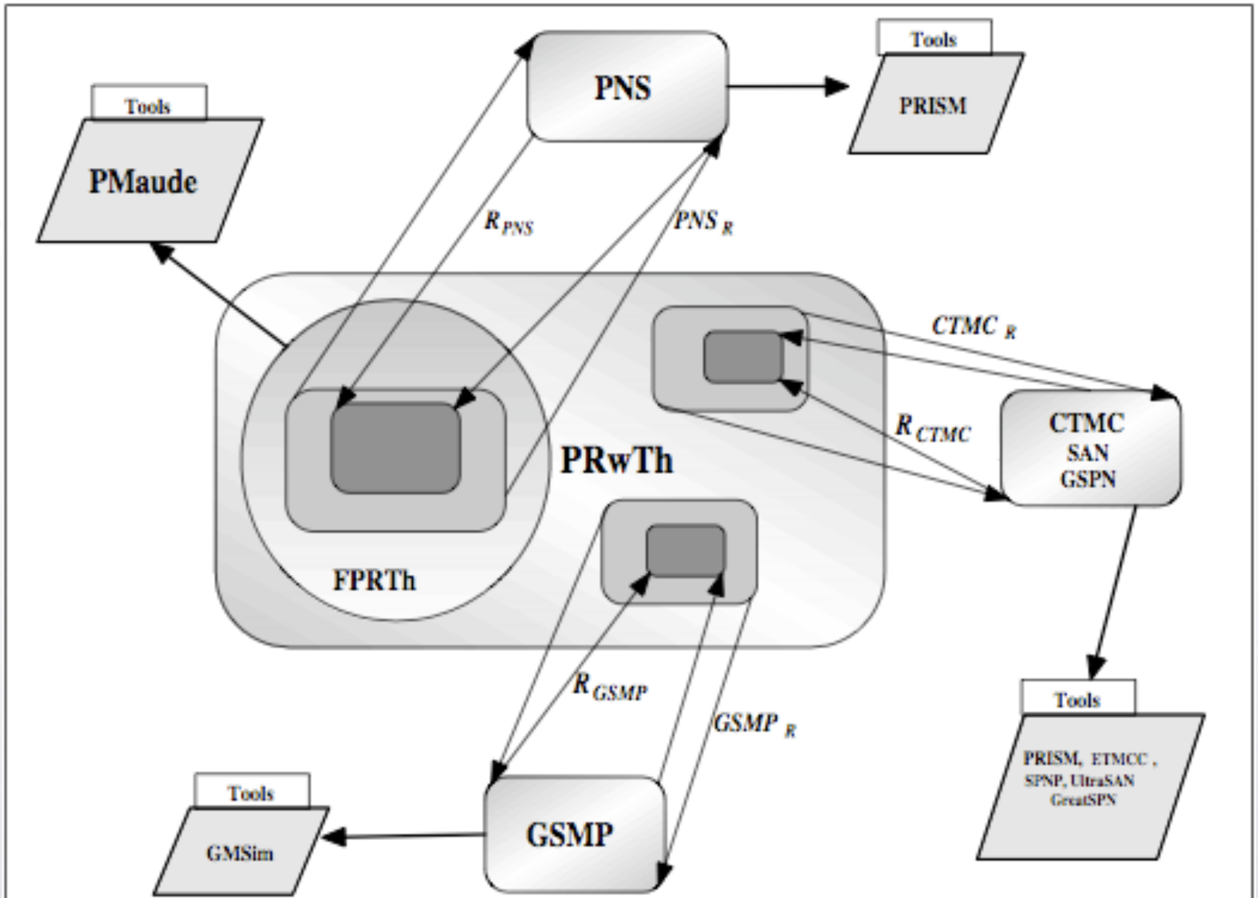
- AER/NCA suite of protocols for reliable, scalable, and TCP-friendly multicast in active networks -- correctness, performance (worst case times).
- OGDC (Optimal Geographical Density Control) wireless sensor network algorithm for picking active nodes
  - Always reach stable/sensing state
  - bound on time to stable state, coverage
- Wide-mouth frog key sharing -- search for matching connections, attacks

Probabilistic Rewriting
&
Maude

Probablistic Rewrite Theory

- PR = ((S,O),E, R, π)
  - ((S,O),E, R) is a rewrite theory
  - π maps rules to probability distribution functions
- prl l : t(***x***) => t'(***x***,***y***) if C(***x***,***y***) with probability ***y*** := π_l(***x***)
- Probablistic Rewriting Temporal Logic
  - $P^q{}_{\#p}\ \varphi$ -- q in {∀,∃}, # in {≤,≥,<,>}
    - probability that φ holds on all/some paths is # p

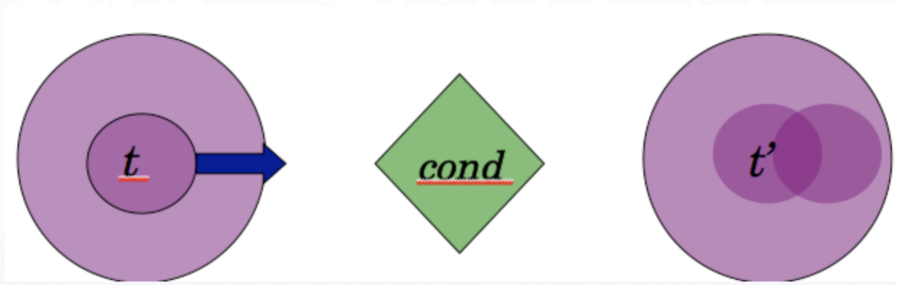
Expressiveness



PNS

PMaude

prl: clock(t,c) ⇒
 if B then clock(t+1, c - c/1000) else broken(t, c - c/1000) fi
  with probability B := BERNOULLI(c/1000) .



crl: clock(t,c) ⇒
 if B then clock(t+1, c - c/1000) else broken(t, c - c/1000) fi
  if B := float(random(seed)/maxRand) < c/1000) .

Analysis methods

- testing -- Monte Carlo simulation
- statistical model checking -- Vesta tool
  - CSL properties
- statistical qualitative analysis: Quatex language
  - E[term] with error bound, confidence

Analyzing TCP/IP SYN Attack

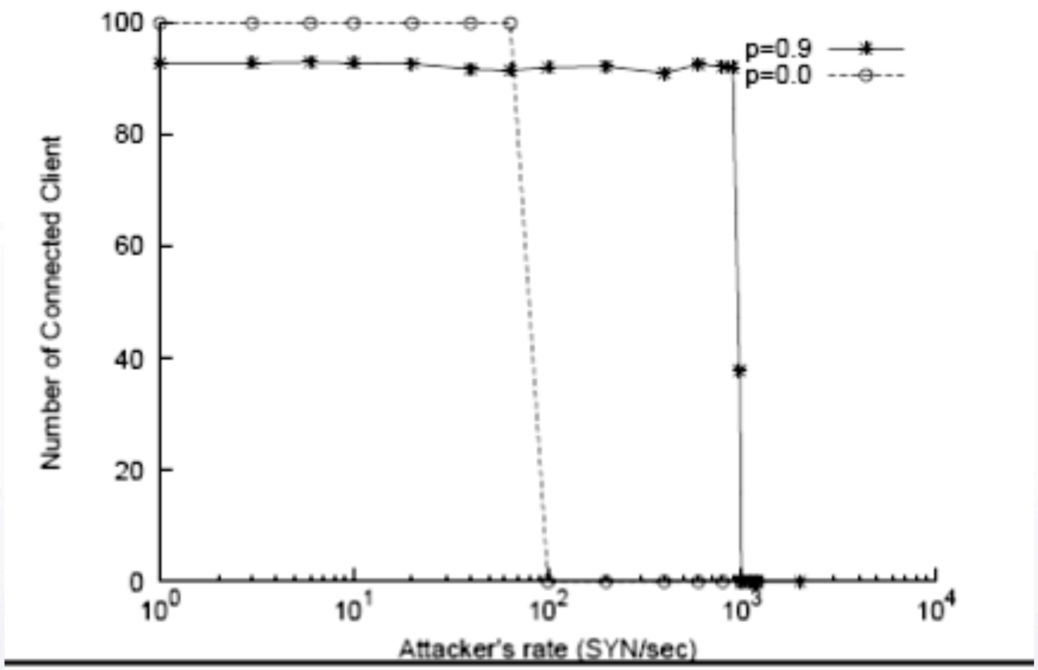
- Problem: attacker fills syn-queue
- Counter measure -- only check fraction p of syn's (client must sent multiple requests)
- Analysis: (for different p)
  - expected number of (of 100) clients that successfully connect
  - probablility that client connects within time t of initiating a request
  - probablility of successfull attack ≤ .01

Statistical MC

- Cache size =10,000
- timeout = 10 seconds
- number of valid senders = 100

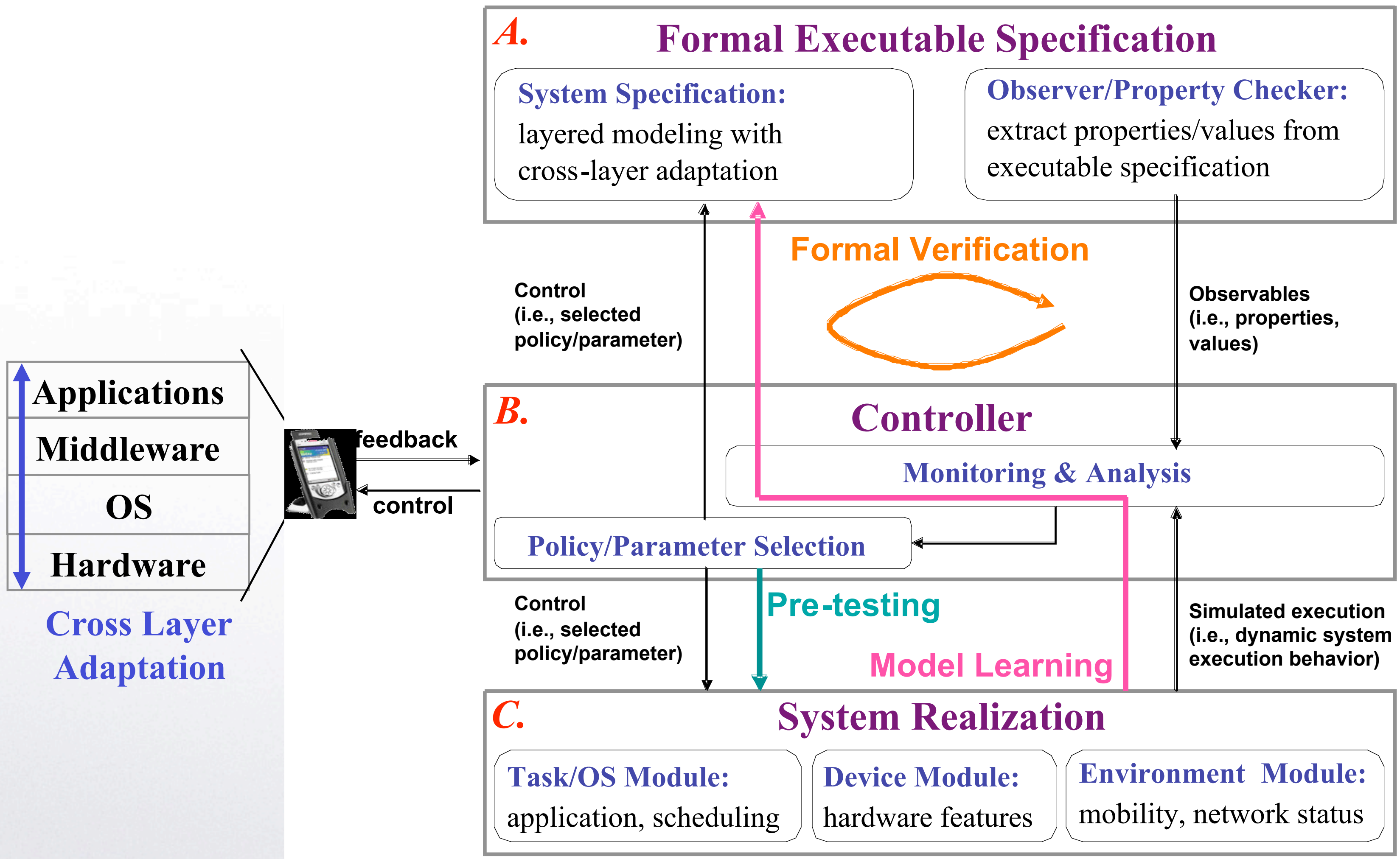
Model-checking $\mathbf{P}_{\leq 0.01}(\Diamond(successful\_attack()))$		X's attack rate (SYNs per second) 1	5	64	100	200	400	800	1000	1200
$p = 0.0$ (No counter-measure)	result	F	F	F	T	T	T	T	T	T
	time (10^2 sec)	47	87	280	605	183	183	182	182	181
$p = 0.9$ (With counter-measure)	result	F	F	F	F	F	F	F	T	T
	time (10^2 sec)	68	75	217	328	896	3102	11727	2281	1781

Quatex Analysis



Expected number of clients out of 100 clients that get connected with the server under DoS attack

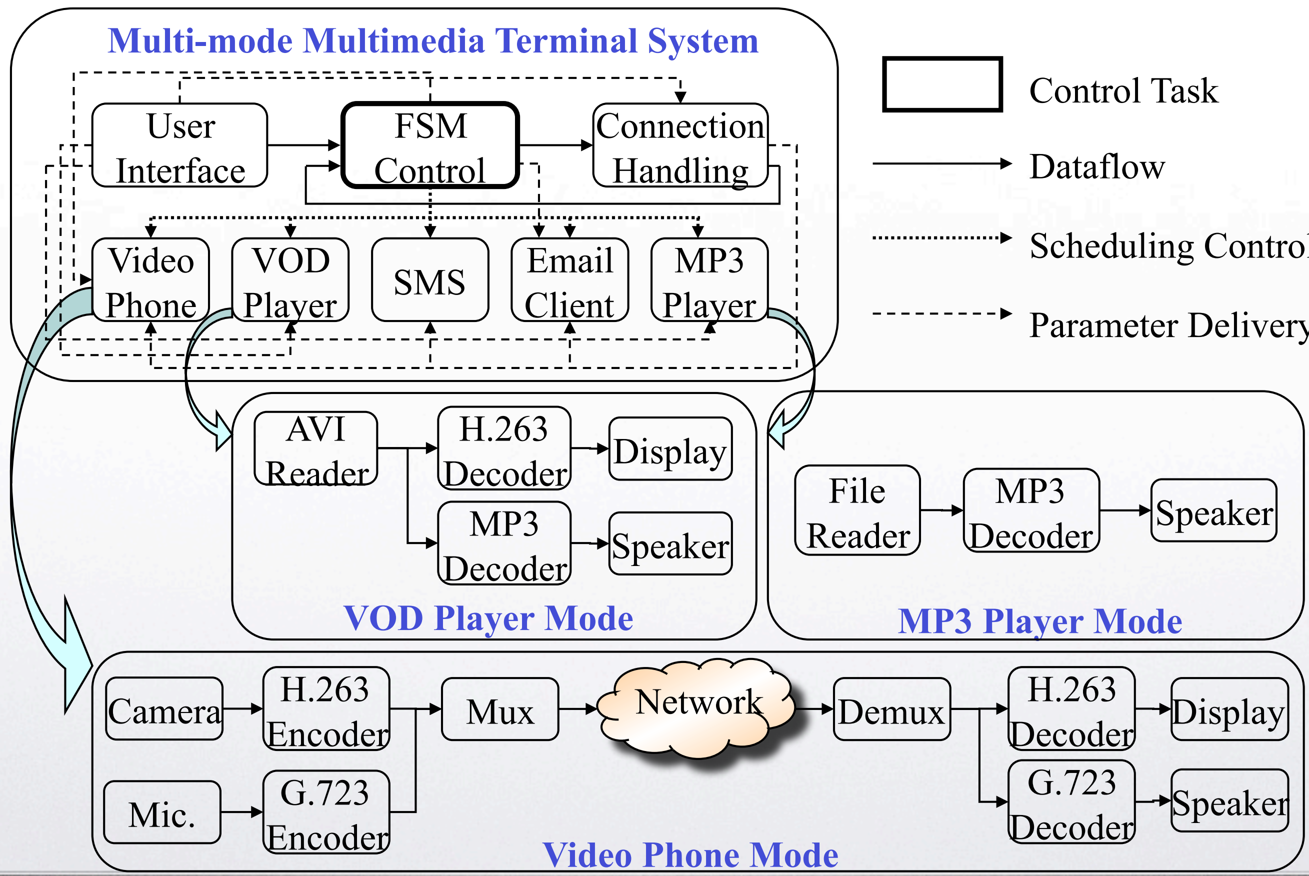
XTune
Cross layer adaptive tuning


Applications
Middleware
OS
Hardware
Cross Layer Adaptation
feedback
control
A. Formal Executable Specification
System Specification:
layered modeling with cross-layer adaptation
Observer/Property Checker:
extract properties/values from executable specification
Formal Verification
Control (i.e., selected policy/parameter)
Observables (i.e., properties, values)
B. Controller
Monitoring & Analysis
Policy/Parameter Selection
Pre-testing
Model Learning
Control (i.e., selected policy/parameter)
Simulated execution (i.e., dynamic system execution behavior)
C. System Realization
Task/OS Module:
application, scheduling
Device Module:
hardware features
Environment Module:
mobility, network status

XTune approach

- System components/layers modeled as objects
- Rules mix time and probability
  - combine ideas of RTMaude and PMaude
- Analysis simplifies/improves ideas of PMaude

Example: Mobile Multimedia



XTune model of video phone

System state -- a clocked configuration

```
{ < CPU: HW | Timer : 0, policy : P,  consumedEnergy : 0.0, ... >
  < pbpair:Application | Timer: 0, accEncTime : 0, consecutiveMiss : 0,  ... >
  < Mobility : NetworkMonitor | Timer : 0, pos : L,  speed : 1, ... >
  < ZoneInfo : Zone | currentDLY : dly, currentPLR : alpha ... >
  < Random : RandomNGen | seed : N >
  ... } in time 999999 .
```

```
crl [tick]: {conf} in time T => {delta(conf, T')} in time (T monus T')
   if T' := mte(conf) /\ T gt T' /\ T' gt 0 .
```

Application execution times, packet arrival times ... sampled from normal and exponential distributions.

Experiments: Statistical MC

Quick detection of problematic situations (e.g., battery expires)

Sequential testing

- Property [probability (battery expires) < 0.1]
- Parameters
  - alpha (false negative) = 0.05, beta (false positive) = 0.05
  - theta (threshold) = 0.1 , delta (indifference region) = 0.01
- 133 traces give H1 accept

Black-box testing also confirms the formula with error of 8.20E-7 with same traces.

Performance

- The run time for each statistical model checking is 10-20 msecs in addition to the sample generation
- a feasible proposition for the on-the-fly adaptation

Experiments: Statistical Analysis

(a) Energy Consumption:
[nSample = 100] Fail to reject Ho (p-value = 0.821)
E[Energy Consumption] = 3.7121E9 ($\alpha = 5.0\%$, $d = 0.036\%$)

(b) Decoder Average Deadline Miss Ratio:
[nSample = 100] Reject Ho (p-value = 0.035)
[nSample = 110] Fail to reject Ho (p-value = 0.194)
E[Decoder Avg Deadline Miss Ratio] = 0.2032 ($\alpha = 5.0\%$, $d = 0.466\%$)

(c) Decoder Maximum Consecutive Lost:
[nSample = 100] Fail to reject Ho (p-value = 0.884)
[nSample = 100] $(d = 0.01053) > (\delta = 0.01)$
[nSample = 110] $(d = 0.01002) > (\delta = 0.01)$
[nSample = 121] $(d = 0.00958) \leq (\delta = 0.01)$
E[Decoder Maximum Consecutive Lost] = 3.2314 ($\alpha = 5.0\%$, $d = 0.958\%$)

(b) The first normality (JB) test fails need more samples
(c) The confidence interval from initial samples is greater than the desired interval
=> need more samples

Summary

- Quantitative analysis in Maude is done by
  - extending basic rewriting with time and probablilities (a built in random number generator)
  - mapping special syntax to core Maude
  - execution, search, and various forms of model checking / statistical analysis

References

Maude

M. Clavel, F. Duran, S. Eker, J. Meseguer, P. Lincoln, N. Martı-Oliet, and C. Talcott. All About Maude – A High-Performance Logical Framework. Springer LNCS Vol. 4350, 2007.

RTMaude

P. C. Olveczky and J. Meseguer. Abstraction and completeness for RealTime Maude. In WRLA 2006, pages 128–153.

P. C. Olveczky, J. Meseguer, and C. L. Talcott. Specification and analysis of the AER/NCA active network protocol suite in Real-Time Maude. Formal Methods in System Design, 29(3):253–293, 2006.

P. C. Olveczky and S. Thorvaldsen. Formal modeling and analysis of wireless sensor network algorithms in Real-Time Maude. In IPDPS 2006.

P. C. Olveczky and M. Grimeland. Formal Analysis of Time-Dependent Cryptographic Protocols in Real-Time Maude. In IPDPS 2007.

References

PMaude

Koushik Sen, Nirman Kumar, José Meseguer and Gul Agha. Probabilistic Rewrite Theories: Unifying Models, Logics and Tools. Technical Report UIUCDCS-R-2003-2347, UIUC, May 2003.

Gul Agha, Michael Greenwald, Carl Gunter, Sanjeev Khanna, Jose Meseguer, Koushik Sen, and Prasanna Thati. Formal Modeling and Analysis of DoS Using Probabilistic Rewrite Theories. In FCS 2005.

XTUNE

M. Kim, M.-O. Stehr, C. Talcott, N. Dutt, and N. Venkatasubramanian. A probabilistic formal analysis approach to cross layer optimization in distributed embedded systems. In FMOODS 2007, LNCS, vol. 4468, pp. 285–300.