Executable Formal Models in Rewriting Logic

Carolyn Talcott
RTA 2015
Formal Executable Models

• For design, prototyping, analysis.

• To clarify ideas, squash insidious bugs early on
  • many bugs / flaws can be found by just formalizing

• To build models to test initial ideas
  • watch it run, poke it, find unexpected order of execution
  • check simple properties by search, symbolic search, model checking
Plan

• About formal systems and Rewriting Logic

• Maude’s formal tools and environments

• Sample formal models
  • in brief
  • in detail

• Wrapup
Formal Modeling
Modeling 101

• What questions do you want the model answer?

• What can you observe/measure?

• What questions do you really want the model answer?

• What does that mean?

• Explain it to a computer!

  • Need a formal representation system
Formal Modeling Methodology

Curator/model builder

data

model checking

S |= \Phi

model

asking questions

rapid prototyping

state space search

impact
A formal model needs a formal system

- Language: to describe things and properties
- Semantics: thing satisfies property
- Reasoning principles: proving/disproving properties of things
- Reflection: to model and reason about models and reasoning
- Executable formal models (model train, airplane, ...)
  - System state: collections of entities
  - State transition rules
  - Execution: application of rules
  - Properties of states (P,Q) and executions
    - ($\phi$: P until Q, eventually P)
- Watch it run, poke it, analyze it
Symbolic analysis -- answering questions

• Forward collection -- upper bound on possible states

• Backward collection -- initial states leading to states of interest

• Search -- for (symbolic) state of interest

• Model checking -- do all executions satisfy $\phi$, find counter example

• Constraint solving -- steady state analysis
About RWL and Maude

A formal representation system and execution environment
Rewriting Logic & Maude

- Rewriting logic is a simple logic designed to model concurrent and distributed systems,
  - System states described by equational theories, behavior described by local rules
- Maude is modeling environment based on rewriting logic, featuring
  - high speed rewriting modulo axioms
  - built in search, model-checking, unification
  - reflection
  - variant generation and variant narrowing
  - rewriting modulo constraints
What is Rewriting Logic?

- A logic for executable specification and analysis of systems, that may be concurrent, distributed, or even mobile.

- A logic to specify other logics or languages

- An extension of equational logic with local rewrite rules to express: concurrent change over time / inference rules: Dual use of rewrite rules

- A rewrite theory plus a term describes a state transition system
  - states can have rich algebraic structure
  - transitions are local and possibly concurrent

- The equational part of a rewrite theory is similar to a term rewriting system (modulo ACI axioms), BUT
  - It is usually desirable for equations to be CR and terminating
  - Rewrite rules are often non-deterministic and non-terminating
Example: A Vending Machine
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
Using the vending machine model: execution and search

• What is one way to use 3 $s?
  
  • Maude> rew $ $ $ .
  
  • result: Marking: a q 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
Using the vending machine model: model checking

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

Model check the assertion that we can't.

Maude>
red modelCheck(vm($ $ $ $ $),[]~(lte4Q U nApples(6))) .
result ModelCheckResult: counterexample(
    {vm($ $ $ $ $),'buy-a'}
    {vm($ $ $ $ q a),'buy-a'}
    {vm($ $ $ q q a a),'buy-a'}
    {vm($ $ q q q a a a),'buy-a'}
    {vm($ q q q q a a a a),'change'}
    {vm($ $ a a a a),'buy-a'}
    {vm($ q a a a a a),'buy-a'}
    {vm(q q a a a a a),deadlock})
Rewriting Logic is Reflective!

- A reflective logic is a logic in which important aspects of its metatheory (entailment relation, theories, proofs) can be represented at the object level in a consistent way.

- This has many applications:
  - Transforming, combining rewrite theories
  - Execution / proof strategies
  - Meta tools: theorem provers, coherence checkers ...
  - Language extensions: object-oriented, real-time, ...
  - Higher-order capabilities in a first-order framework
  - Model of reflection for concurrent objects
  - Domain specific assistants
Reflection example: A simple strategy interpreter

Simple strategy: a list of rule (ids) to apply, in order.

fmod METAREWRITE-LIST is
  inc MY-META.
  var M : Module . vars T T' : Term .
  var res : Result4Tuple? .
  var rid : Qid . var ql : QidList .

  op metaRewList : Module QidList Term -> Term .
  eq metaRewList(M,nil,T) = T .
  ceq metaRewList(M,rid ql,T) = metaRewList(M,ql,T')
    if res := metaXapply(M,T,rid,none,0,unbounded,0)
    \ T' := if res :: Result4Tuple
    then getTerm(res)
    else T fi .
endfm
Reflection: Using the simple strategy interpreter

Maude> red metaRewList(['VENDING-MACHINE],
               'change 'buy-a,
               '___['q.Coin,'q.Coin,'q.Coin,'q.Coin]) .

result GroundTerm: '___['q.Coin,'a.Item]

Maude> red metaRewList(['VENDING-MACHINE],
               'buy-a 'change,
               '___['q.Coin,'q.Coin,'q.Coin,'q.Coin]) .

result Constant: '$.Coin
A sampling of formal environments and tools
The Maude Formal Environment (MFE)

- Integrates tools for reasoning about Maude specifications:
  - Maude Termination Tool (MTT),
  - Church-Rosser Checker (CRC),
  - Coherence Checker (ChC),
  - Sufficient Completeness Checker (SCC),
  - Maude's Inductive Theorem Prover (ITP).

- http://maude.lcc.uma.es/MFE/
Real time Maude

- A language and tool for formal specification and analysis of real-time and hybrid systems.
  - Implemented in (full) Maude
  - timed rewriting and search
  - time-bounded and unbounded LTL and timed CTL (TCTL) model checking.
- Time sampling strategies for execution and analysis proved sound for a large class of specifications.
- Ptolemy II: graphical modeling/simulation tool for embedded systems
  - RT Maude is a fully integrated plugin
- Synchronous AADL (industry standard for embedded systems modeling)
  - Eclipse plug-in for OSATE AADL modeling environment
- [http://heim.ifi.uio.no/peterol/RealTimeMaude/](http://heim.ifi.uio.no/peterol/RealTimeMaude/)
The K framework

- A framework for formal language definition (syntax and semantics) and automatic generation of language specific tools

- Parser, Interpreter, Compiler

- Semantic debugger

- Test-case generation

- Symbolic Execution

- Model checker

- Deductive program verifier

- Application to C, Java, JavaScript, Python, ....

- http://www.kframework.org/
Maude NPA

• A tool for reasoning about cryptographic protocols
  • If Bob finished did Alice also finish?
  • Did Eve learn the secret?
• User definable (in Maude)
  • crypto algebra
  • honest player moves (strands)
  • attacker model, attack patterns
• Backwards narrowing from attack allows unbounded sessions
• Search pruning techniques for managing state space.

• [Link](http://maude.cs.uiuc.edu/tools/Maude-NPA/)
Application sampling
Uncovering security flaws in GUI logic

- Formalization of GUI logic and user interaction invariants
  - abstract document trees
  - abstract interaction sequences
- Based on in depth study of browser code
- Systematic exploration lead to identifying
  - 9 status bar spoofing patterns
  - 4 address bar spoofing patterns
- All confirmed by IE developers (and fixed)
Analysis of active network protocols

Active Error Recovery / Nominee based Congestion Avoidance (AER/NCA)
a suite of protocols to achieve adaptive reliable multicast

Key AER/NCA components
• (RS) Repair Service: ensure that each packet is eventually received by each receiver in the multicast group.
• (RC) Rate Control: adjust packet sending rate, according to loss rate
• (NOM) NOMinee receiver: tries to find the worst receiver, based on the loss rates and the distance to the sender.

Modeling challenges:
• Time-sensitive behavior, timers, ordering
• Delay and delay estimation
• Resource-sensitive behavior, resource contention
• Capacity, latency, congestion/cross-traffic, buffering
• Analyze
  • correctness and performance as critical metrics
  • component-wise and aggregate behavior
What was learned

Formalization started with detailed but informal use case analysis provided by the developers of AER/NCA

Analysis by execution:
• The Repair Service reaches an Error state
• Inspection of the rules lead to discovery of the rule introducing the error state
  • bound on NAK count exceeded
• Examining intermediate states lead to discovery of the cause of the faulty behavior
  • repair server has dropped the repair packet and lost ability to recover it

Analysis by model-checking
• Property: If NOM says there is a nominee, then some receiver has its nominee flag set to True.
  • Only a receiver with nominee flag True acknowledges data packets.
  • Unacknowledged data packets may lead to rate control problems
• Model-checking found a counter example.
And a few more

- XTune: compositional cross-layer optimization framework, using formal executable models of resource usage

- Border Gateway Protocol policy analysis: termination of route discovery

- Application level DOS attack mitigation
  - Design, evaluated in Maude, implemented with (25% >> 90% service)

- Cyber physical protocols: in between ticks attacks due to discrete vs continuous time (rewriting modulo SMT)

- Document logic: analysis of processing specification ala security protocols

- Clinical Trials assistant: specification of trial protocol and checking conformance
Some executable specifications in more detail

Secure Proxy Toolkit (SPTK)

Pathway Logic
SPTK
Problem: Providing Remote Services

• Publication and discovery

• Remote messaging

• QoS Requirements
  • Transparency
  • Security
    • Integrity -- Getting the right / expected service
    • Access control — protecting service
  • Confidentiality
Solution: Service Proxy Toolkit (SPTK)

SSPTK -- a Java implementation of a secure SPTK
Model input — message diagrams
Formal model of architecture developed with plug able components for different security levels and attack models.

<table>
<thead>
<tr>
<th></th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 1</td>
<td>--</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 2</td>
<td>--</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 2a</td>
<td>--</td>
<td>--</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 3</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2.1</th>
<th>2.2</th>
<th>2.3</th>
<th>2.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 1</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 2</td>
<td>+</td>
<td>--</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Level 2a</td>
<td>+</td>
<td>--</td>
<td>--</td>
<td>+</td>
</tr>
<tr>
<td>Level 3</td>
<td>+</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

Analysis with Attacker as network
1.1 attacker sees/modifies client data
1.2 attacker replaces registered proxy (clients gets wrong service)
1.3 illegal or unauthorized service call
1.4 client imposter succeeds

Analysis with Compromised Registry
2.1 client app can get proxy to requested/registered service (sanity check)
2.2 client app accepts proxy to attacker service
2.3 client app accepts wrong proxy
2.4 service integrity violated
Pathway Logic

Executable models of cellular processes

http://pl.csl.sri.com
Pathway Logic (PL) Goals

- Understanding how cells work
- Formal models of biomolecular processes that capture biologist intuitions and can be executed
- Tools to organize and analyze experimental findings, carry out gedanken experiments, and discover/assemble execution pathways
- New insights into the inner workings of a cell.
ErbB network cartoon—biologists review model

PL from 1k feet

Key components

- Representation system
  - controlled vocabulary
  - datums
  - rules
- Curated datum knowledge base (KB) and search tool
- Evidence based rule networks
  - STM, Protease, Mycolate, GlycoSTM
- Executable models
  - generated by specifying initial conditions and constraints
  - query using formal reasoning techniques
- Visualize and browse subnets
sort HrasSort .
subsort HrasSort < RasS < BProtein .

op Hras : -> HrasSort [ctor metadata ( (spnumber P01112) (hugosym HRAS) (synonyms "GTPase HRas"
                           "Transforming protein p21"
                           "v-Ha-ras Harvey rat sarcoma viral oncogene homolog"
                           "Harvey murine sarcoma virus oncogene"
                           "H-Ras-1"
                           "c-H-ras"
                           "HRAS1"
                           "RASH1"
                           "RASH_HUMAN"))]

op Rass : -> RasS [ctor metadata ( (category Family) (members Hras Kras Nras))]

op Pi3k : -> Composite [ctor metadata "(
                          (subunits Pik3cs Pik3rs)
                          (comment "PI3 Kinase is a heterodimer of:
                                    "a p110 catalytic subunit: Pik3ca, Pik3cb, Pik3cd or Pik3cg"
                                    "a p85 regulatory subunit: Pik3r1, Pik3r2, or Pik3r3")]]
Rule Knowledge Base (RKB) — A Rewrite Theory

• describe local change

• specify context shown to be required

• linked to evidence file -- all the datums relevant to the rule

rl[529.Hras.irt.Egf]:

< Egf : [EgfR - Yphos], EgfRC > < [gab:GabS - Yphos], EgfRC >
< [hrasgef:HrasGEF - Yphos], EgfRC > < Pi3k, EgfRC > < [Shp2 - Yphos], EgfRC >
< [Hras - GDP], CLi >

=>

< Egf : [EgfR - Yphos], EgfRC > < [gab:GabS - Yphos], EgfRC >
< [hrasgef:HrasGEF - Yphos], EgfRC > < Pi3k, EgfRC > < [Shp2 - Yphos], EgfRC >
< [Hras - GTP], CLi >

*** ~/evidence/Egf-Evidence/Hras.irt.Egf.529.txt

• Symbolic rules represent a family of rules using sorted variables

• gab:GabS is a variable standing for Gab1 or Gab2, hrasgef:HrasGEF is a variable for any of several HrasGEFs (enzymes to exchange GDP for GTP)
Where do rules come from?

They are inferred from experimental findings.

These are collected using a formal data structure called **datums**

**The Elements of a Datum**

- **Subject**: xHras[tAb]
- **Assay**: GTP-association[BDPD] is increased irt Egf (5 min)
- **Environment**: cells: VERO<xHras><xGab1> in BMLS
- **Extra**: inhibited by: xGab1(Y627F) [substitution]
- **Source**: source: 15574420-Fig-5a
Example signal propagation (using Petri Nets)

Sos1Dish = rule1 => Sos1Dish1 = rule5 => Sos1Dish2 = rule13 => Sos1Dish3

Ovals are occurrences -- biomolecules in locations (aka places).
Dark ovals are present in the current state (marked).
Squares are rules (aka transitions).
Dashed edges connect components that are not changed.
What can be done with an executable RKB?

• Generate a model, for example, of response to some stimulus
  • Define initial state -- cell components and additions -- experimental setup
  • Forward collection gives a network of all the possibly reachable rules
• The Signal Transduction Model (STM) RKB comes with > 30 models of response (of a resting cell) to different treatments (Egf,Insulin,Tnfa,Tgfb, Lps (bug bit), Serum, ....)
• From a model you can
  • generate a subnet relevant to a specific outcome (Erk activated in the nucleus)
  • find an execution path—model-checking the assertion that no path exists
  • carryout in silico knockouts
  • compare nets
  • explore connections up/down stream
The subnet of the Egf model for activating (GTPing) \textbf{Hras}. (Represented as a Petri net.)
Comparing two pathways
Symbolic analysis -- answering global questions

- RMP -- the set of all reaction minimal paths from an initial state to a given set of goals

- From this set we can compute
  - Essential transitions – reactions that are in all pathways to an output.
  - Used places – biochemical species that are in at least one pathway.
  - Knockouts – biochemical species that are in all pathways to an output.
  - Multisignal cellular responses – at least one pathway to an output has more than one stimulus.

- In the Hras subnet
  - 6 execution pathways (3 using Sos1, 3 using RasGrp3 — GEFs)
  - 20 double knockouts (from 4 protein pairs)
Wrapping up
Summary

- Why executable models
- RWL/Maude as modeling environment
- Modeling elements
  - Representation system
  - Model building: systematic, clear thinking!
- Analysis methods
  - Simple execution and search
  - Forwards and backwards reasoning
  - Model-checking
  - Symbolic execution/search
  - Search modulo constraints
- Designed vs natural systems
  - finding bugs vs understanding systems
Looking forward

• Advancing symbolic timed behavior

• Rewriting/search modulo mixed constraints

• Adding soft constraints
  • uncertainty, partial knowledge
  • resource limitation
  • degree of goal satisfaction (not binary)

• Symbolic quantitative reasoning
  • relative rates of processes
  • competition between processes

• **Big Challenge**: Automation of model building
Thanks to

- the (extended) Maude team
- the Pathway Logic team
- the Symbolic Systems Technology group
- many other collaborators ….

Thanks for listening.