Formal specification and analysis of RBP in Maude

Jose Meseguer[1], Carolyn Talcott[2]

[1]Computer Science Laboratory
SRI International, Menlo Park, CA

[2]Computer Science Department
Stanford University, Palo Alto, CA

[with Grit Denker, Peter Ölveczky, and the UCSC AN team]

Why

- Increasing importance, criticality, and complexity of communications software $\leadsto$ application of formal methods in order to gain high assurance about correctness
- Dynamically adaptive software for active networks increase difficulties
- Mathematical modelling of different aspects and stages of system development helps in "getting the right thing in the right way".

What

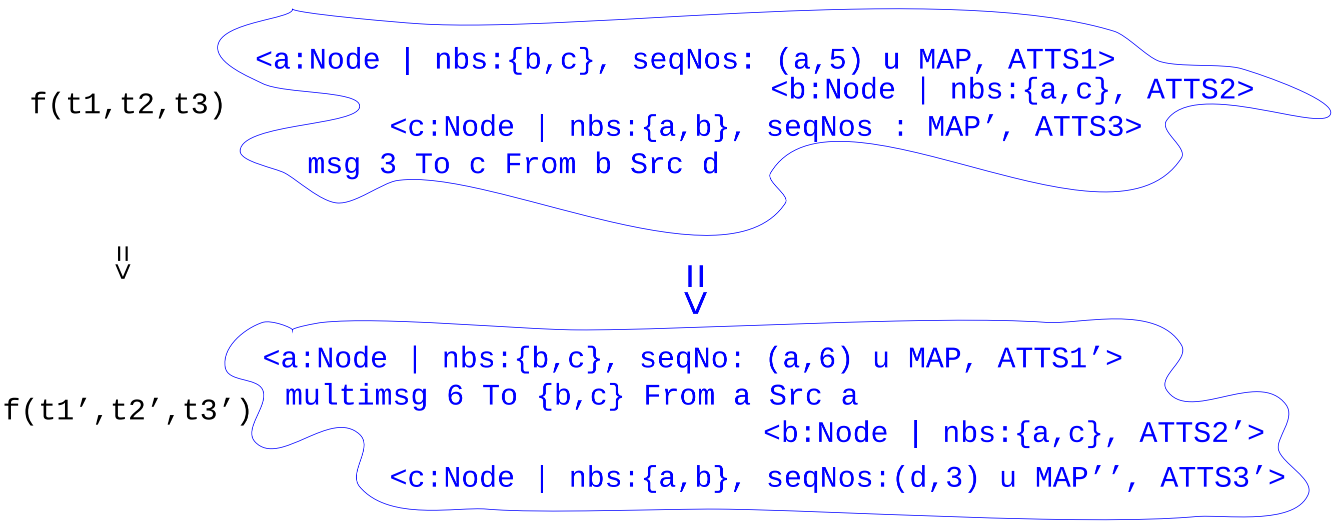
- Formalization and analysis of a new reliable broadcasting protocol (RBP) currently under development at the University of California at Santa Cruz (UCSC) using Maude
- Results:
  - many corrections and improvements to the original design
  - formal, executable specification
  - symbolic execution and formal analysis revealed many bugs and inconsistencies
  - cost efficiency: early in the design process, before implementation

Introducing Maude

- Supports concurrent and distributed computation in a natural way
  - Systems can be synchronous, asynchronous, or hybrid, and can communicate through point-to-point, multicast, broadcast, or combinations
- Supports structuring of distributed systems through domains, firewalls, dynamic distributed services, etc.
- Supports reflective capabilities such as monitoring, management, dynamic adaption, and composable services

Rewriting Logic

- Static structure of the system described as an algebraic data type: `f(t1,...,tn)`
- Dynamics of a system is described by rewrite rules:
  `rl [l] :  t => t'`
- Concurrent transitions!



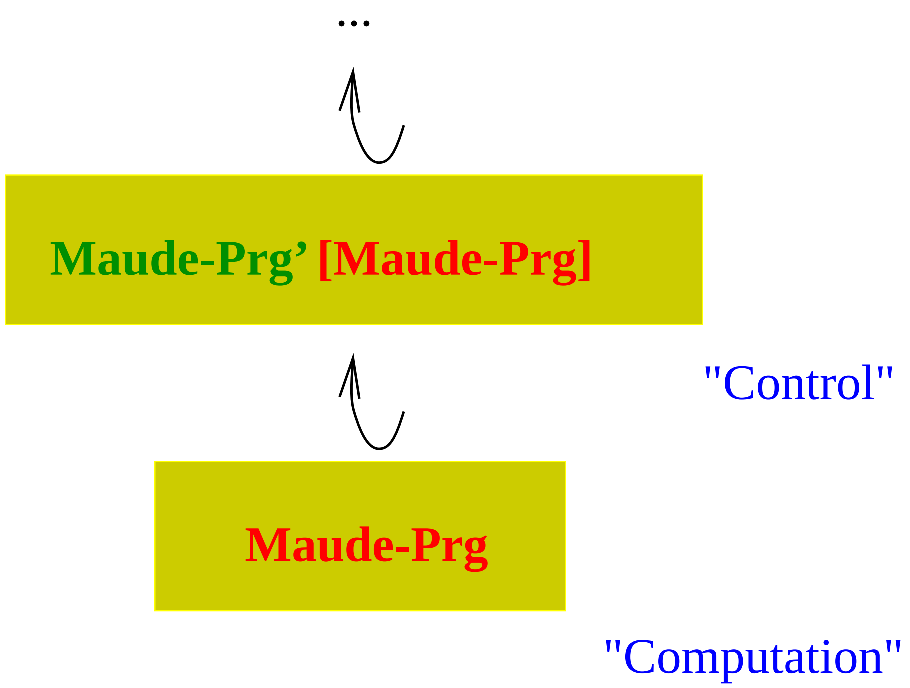
Analysis Techniques

Maude supports a wide spectrum of analysis capabilities:

- Executable specification and testing of specifications
- Model checking and search
- Symbolic simulation
- Theorem proving

How is this done?

Reflection



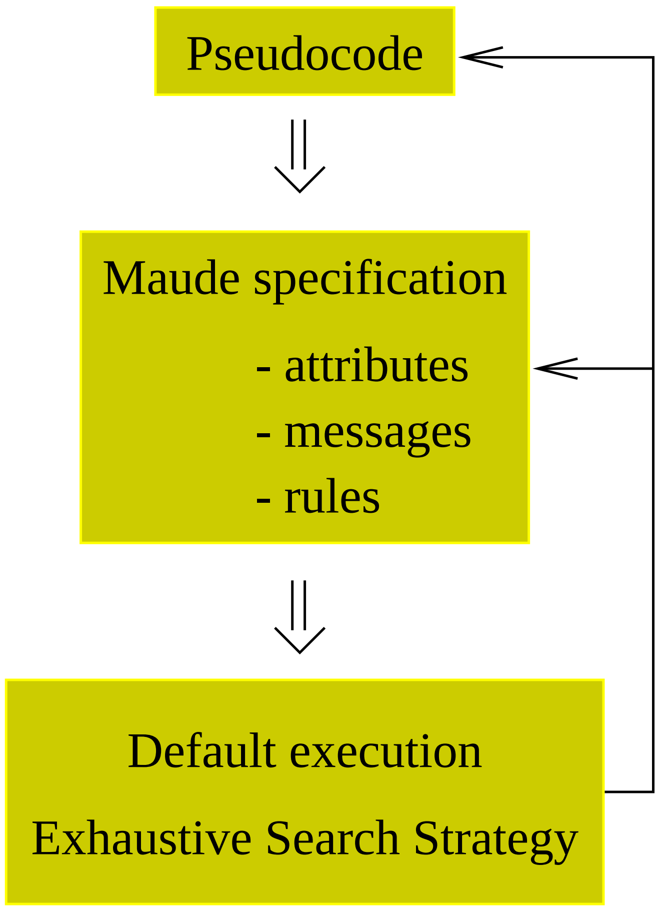
E.g., `Maude-Prg’`
- manipulates execution order of rules in `Maude-Prg`
- defines strategies
- implements model-checker
- implements theorem-prover

Encouraging Experience

Collaboration with other teams in the Active Nets program and other researchers:

- Reliable Broadcast Protocol Design (UCSC (team 4))
- Active Net algorithms (UPenn (team 6))
- Cryptographic Protocol Analysis (with G. Denker (SRI))
- Semantics of CAPSL (J. Millen and G. Denker (SRI))
- Composable Communication Services (team 4)
- Interoperation of Heterogeneous Software Architectures (EDCS)

Overview of Method



RBP

Pseudocode

N^i = set of direct neighbors

ST^i_j = state (active/passive) of node i wrt to source j

SQ^i_j = sequence number

...

Start (i=j):

source node j checks if ST^j_j = passive. If yes, then

- send latest msg for source j to every k in N^j
- ST^j_j := active
- SQ^j_j := SQ^j_j + 1
- ST^j_{jk} := active for all k in N^j
- s^j_j := j

Maude specification

attributes

```
class Node | nbs : OidSet,
             states : Map[Oid,Status],
             seqNos : Map[Oid,Int],
             ...
op active : -> Status .
...
```

messages

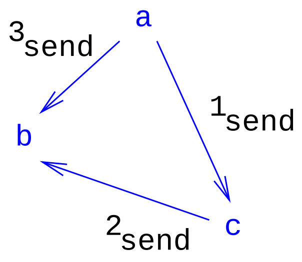
```
msg msg_To_From_Src_ :
    Number Oid Oid Oid -> Msg .
msg multimsg_To_From_Src_ :
    Number OidSet Oid Oid -> Msg .
```

conditional rule

```
crl [Start] :
 < J:Node | nbs : OIDSET,
            states : (J,passive) u MAP,
            seqNos : (J,N) u MAP’
            ... >
 => < J:Node | states : (J,active) u Map,
               seqNos : (J, N+1) u MAP’,
               ... >
    multimsg N+1 To QIDSET From J Src J
 if OIDSET =/= mt .
```

Error: Deadlock

Executing the specification using exploratory strategy delivered a deadlock.




```
c cannot ack a (needs ack from b)
b cannot ack c (needs ack from a)
a cannot ack b (needs ack from c)
```

Original pseudocode rules:

- notation: neighbors = parent + children + siblings
- ack "siblings" after receipt of acks from all children
- ack "parent" after receipt of acks from all siblings and children
- *Node* `a` *is both "grandparent" and "sibling" to node* `b`

RBP – Value Added

Original pseudocode informal specification

- had mistakes and inconsistencies,
- did not consider some cases,
- did not make explicit some key assumptions,
- was not executable.

Maude specification

- found deadlocks and unexpected behaviors,
- made explicit all key assumptions,
- is complete and executable,
- allows systematic exploration of all behaviors, and
- is amenable to formal verification.

Conclusion

- Maximum benefit via early insertion of formal modeling
- Simplicity and intuitive appeal of the formalism "Rewriting Logic"
- Executability of Rewriting Logic: simulation, testing, debugging
- Formal analysis: highly concurrent and nondeterministic behavior of communication protocols becomes feasable