Towards a Maude Formal Environment

Francisco Durán[1] Camilo Rocha[2] José M. Álvarez[1]

[1]Universidad de Málaga
[2]University of Illinois at Urbana-Champaign

Festschrift Carolyn Talcot
November 2011
Menlo Pak, CA

Maude's formal environment

Sufficient completeness

Sort decreasingness

Termination

Confluence

Maude

Coherence

Execution

Theorem proving

Model checking

Reachability analysis

Maude's formal environment: part of Maude or around it

Sufficient completeness

Sort decreasingness

Termination

Confluence

Maude

Coherence

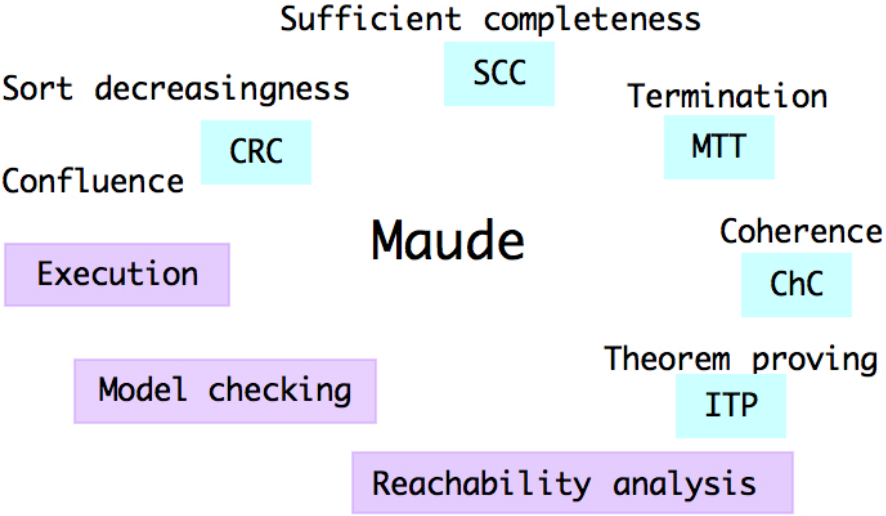
Execution

Model checking

Theorem proving

Reachability analysis

Maude's formal environment: tools around Maude



Maude and its formal environment (MFE)

- Maude is a declarative language and system based on rewriting logic in which computation corresponds to efficient deduction by rewriting.
- Several tools for verifying properties of Maude specifications available.
- These tools work in isolation, making it inconvenient to switch between their environments and difficult to exchange data between them.
- MFE is an executable and highly extensible software infrastructure within which a user can interact with several tools to mechanically verify properties of Maude specifications.
- In MFE, tool interoperability allows for discharging proof obligations of different nature without switching between different tool environments.

Motivation

The Example of Readers and Writers

We want to check that it is never the case that

(i) more than one writer or

(ii) writers and readers

share a critical resource at the same time.

```
fmod MNAT is
  sort MNat .
  op 0 : -> MNat [ctor] .
  op s : MNat -> MNat [ctor] .
endfm
```

```
mod READERS-WRITERS is
  protecting MNAT .
  sort Config .
  op <_,_> : MNat MNat -> Config [ctor] .
  vars R W : MNat .
  rl [wrt+] : < 0, 0 > => < 0, s(0) > .
  rl [wrt-] : < R, s(W) > => < R, W > .
  rl [rdr+] : < R, 0 > => < s(R), 0 > .
  rl [rdr-] : < s(R), W > => < R, W > .
endm
```

The Maude Formal Environment (MFE)

An executable formal specification in Maude within which a user can interact with tools to mechanically verify properties of Maude specifications:

- It has been designed to be easily extended with tools having heterogeneous designs.
  - It currently offers five tools.
- It implements a mechanism to keep track of pending proof obligations.
- Its tool interoperability allows for discharging proof obligations of different nature without switching between different tool environments and presents the user with a consistent user interface.
- It allows the execution of several instances of each tool.

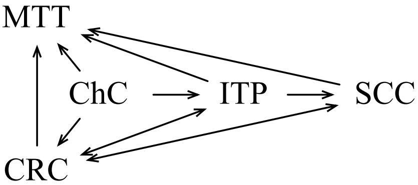
Tool Overview

In the current version of MFE one can interact with the following tools:

- MTT *Maude Termination Tool*
  termination of equational and rewrite specifications
- SCC *Sufficient Completeness Checker*
  sufficient completeness and freeness of equational specifications, and deadlock of rewrite specifications
- CRC *Church-Rosser Checker*
  ground confluence and sort-decreasingness of equational specifications
- ChC *Maude Coherence Checker*
  ground coherence of rewrite specifications
- ITP *Inductive Theorem Prover*
  inductive properties of equational specifications

Tool-dependency Graph in MFE

One important aspect in the integration task is the interaction complexity due to the nontrivial dependencies among tools



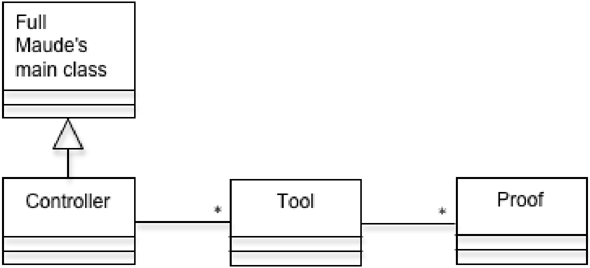
MFE Design Overview

- MFE is modeled in Maude as an interactive object-based system where
  - tools are objects,
  - the communication mechanism is message passing, and
  - user interaction is available through Full Maude.
- Integration and interoperation of tools within MFE is module-centric given that its main purpose is to support formal analysis of Maude modules.
- Although some classes and functionality are provided in MFE, it imposes no constraint on how each tool should model its particular domain or maintains its internal state.
- The use of patterns such as the *model-view-controller* pattern allows us to maximize reuse and simplify interaction and addition of further tools.

Main Classes

The object-oriented model of MFE consists of three main classes

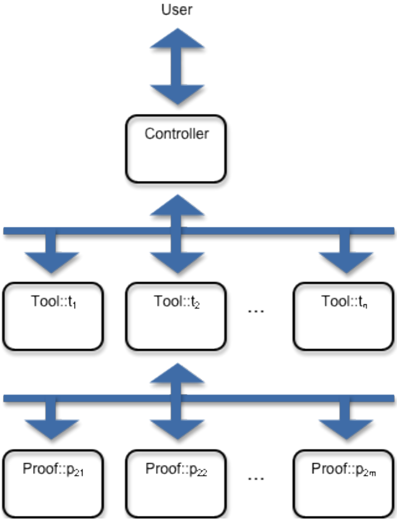
- `Proof` proof objects that keep the state of specific proof requests
- `Tool` tool objects that keep the life-cycle of proof objects
- `Controller` provides a centralized entry point for handling user request



User Interaction

The user interacts with the environment via commands

- each command is encapsulated as a message in the object configuration
- each tool object and the controller object have a module defining the signature of commands it can handle



READERS-WRITERS Church-Rosser

```
Maude> (select tool CRC .)
CRC has been set as current tool.

Maude> (check Church-Rosser .)
Church-Rosser check for READERS-WRITERS
    There are no critical pairs.
    The specification is confluent.
    The module is sort-decreasing.
    Success: The module is therefore Church-Rosser.
```

READERS-WRITERS ground coherent

```
Maude> (select tool ChC .)
ChC has been set as current tool.

Maude> (check coherence .)
Coherence checking of READERS-WRITERS
    All critical pairs have been rewritten and no rewrite with rules
    can happen at non-overlapping positions of equations left-hand sides.
    The termination and Church-Rosser properties must still be checked.
```

READERS-WRITERS ground coherent (proof obligations)

```
Maude> (submit .)
The Church-Rosser goal for READERS-WRITERS has been submitted to CRC.
The termination goal for the functional part of READERS-WRITERS has been
    submitted to MTT.
Success: The functional part of module READERS-WRITERS is terminating.
Church-Rosser check for READERS-WRITERS
    There are no critical pairs.
    The specification is confluent.
    The module is sort-decreasing.
    Success: The module is therefore Church-Rosser.
The functional part of module READERS-WRITERS has been checked terminating.
The module READERS-WRITERS has been checked Church-Rosser.
Success: The module READERS-WRITERS is coherent.
```

Some properties of READERS-WRITERS

Mutual exclusion? First, define an abstraction and proof its correctness.

```
mod READERS-WRITERS-PREDS is protecting READERS-WRITERS .
  ops mutex one-writer : Config -> MBool [frozen] .
  vars M N : MNat .
  eq mutex(< s(N), s(M) >) = false .
  eq mutex(< 0, N >) = true .
  eq mutex(< N, 0 >) = true .
  eq one-writer(< N, s(s(M)) >) = true .
  eq one-writer(< N, s(0) >) = true .
  eq one-writer(< N, 0 >) = true .
endm

mod READERS-WRITERS-ABS is including READERS-WRITERS-PREDS .
  eq [abs] : < s(s(N:MNat)), 0 > = < s(0), 0 > .
endm
```

We need to check:

- the equations being ground confluent, sort-decreasing, and terminating;
- the equations being sufficiently complete; and
- the rules being ground coherent with respect the equations.

Sufficient completeness of READERS-WRITERS-ABS

```
Maude> (select tool SCC .)
SCC has been set as current tool.

Maude> (scc READERS-WRITERS-ABS .)
Checking sufficient completeness of READERS-WRITERS-ABS ...
To complete the proof the specification must be proved ground
sort-decreasing and weakly-terminating.

Maude> (submit .)
The sort-decreasingness goal for READERS-WRITERS-ABS has been submitted
    to CRC.
The termination goal for the functional part of READERS-WRITERS-ABS has
    been submitted to MTT.
Success: Module READERS-WRITERS-ABS is sort-decreasing.
Success: The functional part of module READERS-WRITERS-ABS is terminating.
Success: Module READERS-WRITERS-ABS is sufficiently complete.
```

Church-Rosser property of READERS-WRITERS-ABS

```
Maude> (select tool CRC .)
CRC has been set as current tool.

Maude> (show state .)
State of the tool:
- Church-Rosser check for READERS-WRITERS :
    There are no critical pairs.
    The specification is confluent.
    The module is sort-decreasing.
    The module is therefore Church-Rosser.
- Church-Rosser check for READERS-WRITERS-ABS :
    All critical pairs have been joined.
    The specification is locally-confluent.
    The module is sort-decreasing.

Maude> (submit .)
The termination goal for the functional part of READERS-WRITERS-ABS has
    been submitted to MTT.
The functional part of module READERS-WRITERS-ABS has been checked
    terminating.
Success: The module READERS-WRITERS-ABS has been checked Church-Rosser.
```

Ground coherence of READERS-WRITERS-ABS

```
Maude> (select tool ChC .)
ChC has been set as current tool.

Maude> (check ground coherence READERS-WRITERS-ABS .)
Ground coherence checking of READERS-WRITERS-ABS
The following critical pairs cannot be rewritten:
  cp READERS-WRITERS-ABS1 for abs and rdr-
    < s(0),0 >
    => < s(#1:MNat), 0 > .
The termination and Church-Rosser properties must still be checked.

Maude> (submit .)
The Church-Rosser goal for READERS-WRITERS-ABS has been submitted to CRC.
The goal for critical pair READERS-WRITERS-ABS1 has been submitted to ITP.
The termination goal for the functional part of READERS-WRITERS-ABS has
    been submitted to MTT.
The module READERS-WRITERS-ABS has been checked Church-Rosser.
The functional part of module READERS-WRITERS-ABS has been checked
    terminating.
```

Ground coherence of READERS-WRITERS-ABS

The ITP does not provide methods to prove the joinability of critical pairs. However, we can carry on a proof by reasoning by cases and using Maude's searching command.

```
Maude> (select tool ITP .)
ITP has been set as current tool.

Maude> (trust .)
Eliminated current goal.
The critical pair READERS-WRITERS-ABS1 has been trusted.
Success: The module READERS-WRITERS-ABS is ground-coherent.
```

Invariants check

```
Maude> (search in READERS-WRITERS-ABS :
         < 0, 0 > =>* C:Config
         such that mutex(C:Config) = false .)
No solution.

Maude> (search in READERS-WRITERS-ABS :
         < 0, 0 > =>* C:Config
         such that one-writer(C:Config) = false .)
No solution.
```

Conclusions

- MFE exploits Maude as a reflective declarative language and system based on rewriting logic.
- MFE is an executable and highly extensible software infrastructure within which a user can interact with several tools to mechanically verify properties of Maude specifications.
- Five important formal analysis tools with highly heterogeneous designs:
  - Maude's Termination Tool,
  - Church-Rosser Checker,
  - Coherence Checker,
  - Sufficient Completeness Checker, and
  - Inductive Theorem Prover.

Future work

- Tools such as
  - Maude's LTL and LTLR Model Checkers, and
  - Maude's Invariant Analyzer Tool

  could be integrated in MFE.
- Handling proof obligations such as those
  - for the protecting and extending importations of modules,
  - for the instantiation of parameterized modules, or
  - the termination and Church-Rosser assumptions for equational simplification.
- A graphical user interface and support for better interoperability will enhance the user experience with the formal environment.