The Maude 2.0 System

M. Clavel F. Durán S. Eker P. Lincoln

N. Martí-Oliet J. Meseguer C. Talcott

Overview

- Maude is a language based on rewriting logic.
- Type system is based on membership equation logic.
- Equations are assumed to be confluent and terminating; used for conventional algebraic specification & functional programming.
- Rewrite rules are assume to be coherent w.r.t. equations (Viry); used to express inference & state change.
- Maude modules themselves form an algebraic data type - metaprogramming; extension via reflection .
- Can be used as a semantic framework to specify & prototype other languages and as a logical framework to represent & mechanize other logics.
- Maude 2.0 is the latest incarnation.

New Term Representation

- Supports AC, ACU, A, AU.
- Based on persistent data structures.
- Reduces the computational complexity of E-rewriting for large subjects and simple patterns.

```
fmod REV-LIST is
  sort List Elt .  subsort Elt < List .
  op nil : -> List .
  op __ : List List -> List [assoc id: nil] .
  ops a b c d e : -> Elt .
  vars E E2 : Elt .  var L : List .
```

New Term Representation (2)

```
  op rev : List -> List .
  eq rev(nil) = nil .
  eq rev(E L) = rev(L) E .

  op rev2 : List -> List .
  eq rev2(nil) = nil .
  eq rev2(E) = E .
  eq rev2(E L E2) = E2 rev2(L) E .

  op rev3 : List -> List .
  eq rev3(nil) = nil .
  eq rev3(L E) = E rev3(L) .
endfm
```

- All three naive list reversal algorithms run in linear time!

Conditions

- Rather than a single equality, conditions now allow a list of *condition fragments*, separated by $/\backslash$.
- 4 types of fragments; last type is for rule conditions only:

$\langle$term$\rangle$ = $\langle$term$\rangle$	equality test
$\langle$term$\rangle$: $\langle$sort$\rangle$	sort test
$\langle$pattern$\rangle$:= $\langle$term$\rangle$	assignment by matching
$\langle$term$\rangle$ => $\langle$pattern$\rangle$	rewrite proof search

- Patterns may have unbound variables that are bound by matching; regular terms are reduced upto strategy.
- Failure of a fragment causes backtracking.

Iter Attribute

- Allows efficient storage, i/o and sort computations for huge towers of unary operator symbols.
- Main application is the efficient implementation of natural numbers using the successor notation.

```
fmod ITER-TEST is
  sorts Even Odd Nat .
  subsorts Even Odd < Nat .
  op 0 : -> Even .
  op s_ : Even -> Odd [iter] .
  op s_ : Odd -> Even [iter] .
endfm

red s_^123456789(0) .
red s_^1234567890(0) .
```

Natural Numbers

- Nats are constructed using successor operator and the *iter* attribute.

```
fmod NAT is
  sorts Zero NzNat Nat .
  subsort Zero NzNat < Nat .
  op 0 : -> Zero [ctor] .
  op s_ : Nat -> NzNat
    [ctor iter special (...)] .
  ...
endfm
```

- Decimal i/o by default.
- Built-in operators very efficient (use GNU GMP).
- Gcd, lcm, mod exp, bitwise ops.

Natural Numbers (continued)

```
red in NAT : gcd(18, gcd(X:Nat, 12)) .

fmod COMBINATORIAL is protecting NAT .
  op _! : Nat -> NzNat .
  op C : Nat Nat -> Nat .

vars N M : Nat .
  eq 0 ! = s 0 .
  eq (s N) ! = s N * N ! .
  eq C(N, M) = N ! quo (M ! * sd(N, M) !) .
endfm

red 1000 ! .
red C(1000, 100) .
```

Other Built-ins

- Integers (constructed from Nats).

```
subsorts NzNat < NzInt Nat < Int .
op -_ : NzNat -> NzInt [...] .
```

- Rationals (constructed from Ints and Nats).

```
subsorts NzInt < NzRat Int < Rat .
op _/_ : NzInt NzNat -> NzRat [...] .
```

- IEEE Floating Point Numbers.
- Strings (using SGI Rope package).
- Comprehensive set of conversion functions.

LTL Model Checker

- Linear Temporal Logic manipulations (simplifications and negative normal form) are done by Maude code.
- The satisfaction of (possibly parameterized) propositions are defined in Maude.
- On-the-fly model checking is done by a built-in operation

```
op modelCheck : State Formula ~> ModelCheckResult
  [special (...)] .
```

- Implementation uses state-of-the-art Buchi automaton construction algorithm and standard double depth first search of the synchronous product for a counterexample.
- LTL Satisfiability solving and tautology checking also provided.

New Meta Level

- Simpler metaterm representation:

  ```
  1.0 + X:Float
  ```

  is meta-represented as

  ```
  '_+_['1.0.FiniteFloat, 'X:Float]
  ```

- Many more descent functions such as metaXapply() (with contexts) and metaSearch().
- Descent functions return sorts of terms.
- Ascent functions to inspect modules in database.
- Much more sophisticated caching at module and operator level.

Term Coloring

- Color (possibly intermediate) results based on reduced flag and constructor status to flag problems.

```
set print color on .
```

reduced, ctor	not colored
reduced, non-ctor, colored below	blue
reduced, non-ctor, no colored below	red
unreduced, no reduced above	green
unreduced, reduced directly above	magenta
unreduced, reduced not directly above	cyan

- Red and magenta denote likely origin of problem, blue and cyan denote secondary damage.

Format Attribute

- Allows control of white-space, color and style for pretty-printing operators.
- Format words are given for each white-space position.

s	space	r	red	R	background red
t	tab	g	green	G	background green
+	increment indent counter	b	blue	B	background blue
-	decrement indent counter	y	yellow	Y	background yellow
i	indent by indent counter	m	magenta	M	background magenta
n	new line	c	cyan	C	background cyan
f	flash	w	white	W	background white
h	hidden	p	black	P	background black
d	default spacing	u	underline	x	reverse video
o	original style	!	bright	?	dim

Format Attribute (2)

```
op while _ do _ od : Bool Statement -> Statement .
  ^     ^ ^  ^ ^  ^
```

```
op let _ := _ : Variable Expression -> Statement .
  ^   ^ ^  ^ ^
```

```
op while _ do _ od : Bool Statement -> Statement
  [format (nir! o r! o++ --nir! o)] .
```

```
op let _ := _ : Variable Expression -> Statement
  [format (nir! o d d d)] .
```

Other New Features

- Profiling.
- Break points (on symbols or labeled statements).
- Rewrite search, dump of rewrite graph.
- Kind level declarations & on-the-fly variable declarations.
- Command line editing (Tecla).
- Frozen attribute.
- Position fair & object-message rewriting.
- Statement attributes: owise, label, nonexec, metadata.
- Optimizations: left-to-right sharing, order sorted discrimination nets, substitution slot coloring.
- More powerful module operations in Full Maude.

Availability

Runs under Unix (all flavors).

Licensed under GNU GPL.

Source tree, manual, examples and binaries for selected architectures available from the new Maude website:

http://maude.cs.uiuc.edu/