An Introduction to Maude

Joe Hendrix

Talk Outline

Maude is many things:

- Program Language
  Can be used for representing sequential and concurrent computation.
- Meta Language
  Maude can be used to write programs that extend Maude and reason about Maude code. An extended version of Maude called *Full Maude* is written in Maude itself.
- Logic
  The core logic is called *Membership Equational Logic* and it has an extension for concurrent systems called *Rewriting Logic*.

Sorts, Subsorts and Kinds

- Maude is a typed language where the types are called *Sorts*.
- If one type is a refinement of another tyoe, it can be declared as a *subsort*. e.g. Nat is a subsort of Int.
- The transitive closure of the subsort declarations forms a connected component called a *kind*. Type checking is really done at the kind level, determining the sort of a term requires a TM.

Operators

- Operators are the building blocks for terms.
- They are used to define both data and functions.

Equations

- Equations define how operations transform.
- Conceptually equations create equivalent classes and substitute one equal term with another. *Rules* (explained later) are used to transform terms in ways that do not necessarily subsitute one term for an equivalent one.
- If the equations are all confluent and terminating, then testing if two terms are equivalent can be done by left-to-right rewriting via the rules into a cannonical form.

Axioms

There are three axioms that can be added as attributes to operator declarations.

- Associativity
- Commutativity
- Identity (left, right, and both)

Memberships

- Memberships place terms into sorts. They are used when simple operator declarations are not powerful enough.

Rules

- Rules are used to model state transformations - particularly concurrent and non-deterministic computations.
- Conceptually, rules are not the substitution of *equals for equals*, but change of state over time.
- There is an extension to Maude called Real-time Maude that allows one to attach time to each rule transformation.
- There is another extension called Probablistic Maude that allows rules to be chosen with probabilities instead of non-deterministically.

Meta Language

- Maude can be used to reason about Maude code, and hence it is a meta language.
- For performance reasons, operations at the metalevel can be performed by decent functions that perform computations at the object level.

Execution Strategy

- The default execution strategy is to first evaluate the arguments of a term fully, then look for equations that apply to the term itself.
- Equations are applied in the order they appear in the module.
- Equations can be tagged with the nonexec strategy so they are not executed by the default strategy.
- Equations can also have an operator-level evaluation strategy defined so that some arguments are lazy or arguments are evaluated in a specific order.

Logic

- Maude has a logic associated with the structure that is used by Maude ITP to prove properties about programs.

Model Checker

- Maude has a builtin on-the-fly LTL Model Checker.
- Can work on any Maude module provided the number of reachable states from the initial state is reasonable.
- Maude is competitive in performance with SPIN despite having a much-more flexible syntax.