Rewriting Systems

http://d3s.mff.cuni.cz

Pavel Parízek

Department of Distributed and Dependable Systems

FACULTY OF MATHEMATICS AND PHYSICS
Charles University
Motivation: executable specifications

- Systematic rewriting based on equations
  
  - Example: list length
  
  - Rewrite systems
    - Theory (background)
    - Practice (Maude)
Substitution

- Signature: the set $V$ of variable names

- Substitution $\sigma : V \rightarrow X$
  - Unifier of $t_1$ and $t_2$ if $\sigma(t_1) = \sigma(t_2)$

- Inductively defined sub-expressions

- Reducible sub-expression $t_1[\beta] // \text{redex}$
  - If $\sigma(\beta) = \sigma(t_2)$
Rewriting rules & systems

- Rule \( r : l \rightarrow p \)
- Application of a rule
  - \( \sigma(t)[\beta \leftarrow \sigma(p)] \)
- Rewriting system: set \( R \) of rules
- Derivation \( t \rightarrow_R u \)
  - Reflexive transitive closure \( \rightarrow_R^* \)
- Irreducible expressions
  - Normal form (canonical)
Properties of rewriting systems

- **Confluence**
  - \[ \forall t, t_1, t_2 \cdot ((t \rightarrow_R^* t_1 \land t \rightarrow_R^* t_2) \Rightarrow \exists u \cdot t_1 \rightarrow_R^* u \land t_2 \rightarrow_R^* u) \]

- **Terminating**
  - Normal form always exists

- **Canonical**
  - Single normal form
Canonical values

- Classes of equivalent terms (expressions)
  - Generated by equations (sentences) in the set $E$

- Canonical representatives of classes

- Canonical values (forms) of expressions
Knuth-Bendix procedure

- **Input:** \( Q = (S, \Sigma, E), \leq \subseteq X \times X \)

- **Algorithm**
  1) \( R := \emptyset \)
  2) if \( E = \emptyset \) then return \( R \) // canonical rewriting system
  3) take any \( t_1 = t_2 \in E \) such that either \( t_1 \leq t_2 \) or \( t_2 \leq t_1 \)
     3a) if \( \exists t_1 = t_2 \) then \( E := E - \{t_1 = t_2\} \)
     3b) if \( t_1 \) and \( t_2 \) not comparable then fail // \( R \) cannot be created
  4) if \( t_2 \leq t_1 \) then \( R := R \cup \{R(t_1) \rightarrow R(t_2)\} \)
  5) if \( t_1 \leq t_2 \) then \( R := R \cup \{R(t_2) \rightarrow R(t_1)\} \)
  6) if \( R(t_1) \neq R(t_2) \) then \( E := E \cup \{R(t_1) = R(t_2)\} \)
  7) continue with step 2
Connection to algebraic specifications

- Equations
  - Simple rewriting semantics (simplification)
  - Left-hand side replaced by right-hand side
Maude

- Web: [http://maude.cs.illinois.edu/w/index.php/The_Maude_System](http://maude.cs.illinois.edu/w/index.php/The_Maude_System)
  - source code, documentation, examples
- Version: 3.3 or newer

Main features
- Functional modules and theories
  - Algebraic specifications
- Numeric and string data types
- Computation (rewriting, equations)
  - membership equational logic
- much more (check the web site)
Maude: installation & running

- **Linux**
  - [http://maude.cs.illinois.edu/w/index.php/Maude_download_and_installation](http://maude.cs.illinois.edu/w/index.php/Maude_download_and_installation)

- **Windows**
  - [http://maude.cs.illinois.edu/w/index.php/Installation_guidelines](http://maude.cs.illinois.edu/w/index.php/Installation_guidelines)

- **Running**
  - `<directory with Maude>\maude.exe`
  - From the command-line in a working directory that contains your input files
Maude: basic commands

1) Prepare specification in a text file
2) run the Maude tool
3) load your input file: `load <file>`
4) apply rewriting on some expression
   `reduce [in <module> :] <expr>`
5) Exit the Maude prompt: `quit`
Maude programs: syntax and semantics

• Functional modules
  - sorts, variables, operations, equations

• Notation for operations: prefix, mixfix

• Comments

• Built-in sorts and modules
  - Bool, NAT, INT, FLOAT, RAT, QID, STRING
Maude programs: syntax and semantics

• Examples
  - Natural numbers (Peano arithmetic)
  - Stack of natural numbers

• Theories

• Conditional equations

• Membership axioms

• Attributes of operations
Maude programs: advanced concepts

- Parameterized modules (generic)
  - Example: generic stack

- Importing modules
  - protecting
  - extending
  - including
Maude programs: there is even more

- Data structures (MAP, ARRAY, others)
- Rewriting rules ("basic", conditional)
- Useful built-in modules (CONFIGURATION)
Literature

• Documentation
  - http://maude.cs.illinois.edu/w/index.php/Maude_Manual_and_Examples

• Maude and Rewriting Logic
  - http://maude.cs.illinois.edu/w/index.php/Some_Papers_on_Maude_and_on_Rewriting_Lo