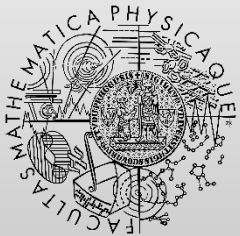


Rewriting Systems

<http://d3s.mff.cuni.cz>



Pavel Parížek



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 $t_1[\beta]$ // 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

- $\forall t, t_1, t_2 \bullet ((t \rightarrow_{R^*} t_1 \wedge t \rightarrow_{R^*} t_2) \Rightarrow \exists u \bullet t_1 \rightarrow_{R^*} u \wedge t_2 \rightarrow_{R^*} u)$

- 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
 - source code, documentation, examples
- Version: 2.7.1 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?title=Maude_download_and_installation
- Windows
 - <http://safe-tools.dsic.upv.es/maude/>
 - <http://moment.dsic.upv.es/>
 - <https://www.cygwin.com/>
- 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)

- Documentation

- [http://maude.cs.illinois.edu/w/index.php?title=Maude Manual and Examples](http://maude.cs.illinois.edu/w/index.php?title=Maude%20Manual%20and%20Examples)
- [http://maude.cs.illinois.edu/w/index.php?title=Maude 2 Primer and Examples](http://maude.cs.illinois.edu/w/index.php?title=Maude%20Primer%20and%20Examples)

- Maude and Rewriting Logic

- M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J.F. Quesada. Maude: Specification and Programming in Rewriting Logic. Theoretical Computer Science, 285 (2), 2002
- <http://maude.cs.illinois.edu/papers/>