Implementation and Performance Evaluation of Multi-Completion with Termination Checking

Formative System Engineering Laboratory,
Graduate School of
Information Science and Technology,
Hokkaido University, Japan

Haruhiko Sato, Masahito Kurihara
Outline

- Backgrounds
  - Equational Logic and Word Problem
  - Term Rewriting System and Knuth-Bendix completion
- Improved Knuth-Bendix completion
  - Multi-completion
  - Constraint-based completion
- Multi-completion based on constraint system
  - Basic idea
  - Some techniques for efficient termination checking
- Conclusion
Word Problem in Equational Logic

Word Problem: given a set of equations \( E \) and an equation \( s = t \), can we derive \( s = t \) with \( E \): i.e., transform \( s \) into \( t \) using \( E \)? (undecidable)

ex. Group Axioms

variables: \( x, y, z \)

signatures: \( f, i, e, a \)

\[
\begin{align*}
\text{derivation of } f(i(a), f(a, a)) &= a \\
\text{deriving: replacing a term with an equivalent term}
\end{align*}
\]

\[
\begin{align*}
\text{whole term matches RHS of (3): } & f(x, f(y, z)) \\
\text{1st argument matches LHS of (2): } & f(i(x), x) \\
\text{whole term matches LHS of (1): } & f(e, x)
\end{align*}
\]

\[
\begin{align*}
f(e, x) &= x \quad \text{(1)} \\
f(i(x), x) &= e \quad \text{(2)} \\
f(f(x, y), z) &= f(x, f(y, z)) \quad \text{(3)}
\end{align*}
\]

Application: automated theorem proving, formal verification of programs
Term Rewriting System (TRS)

- **Term Rewriting System (TRS)**
  - a set of **rewrite rules** $s \rightarrow t$, directed equations
  - repetition of rewriting represents computation
  - result of computation is called **normal form**
  - the way of applying rules is nondeterministic

- Two important properties on TRS
  - **Termination**: no infinite rewrite sequence exists
    - computation always stops (can obtain normal form(s))
  - **Confluence**: normal form is always unique (if exists)
    - above example is not confluent
Completion Procedure

- **Completion Procedure** [Knuth, Bendix: 1970]
- proposed as procedure to solve word problem efficiently
- Given a set of equations $E$ and a reduction order $>$,
generates terminating and confluent (convergent) TRS
- reduction order ensures termination of generated TRS
- convergent TRS can **solve word problem** in $E$
- $s = t$ iff $\text{normalform}(s) = \text{normalform}(t)$

ex. Completion of Group Axioms

- $f(e, x) = x$
- $f(i(x), x) = e$
- $f(f(x, y), z) = f(x, f(y, z))$

\[ f(e, x) \rightarrow x \]
\[ f(x, e) \rightarrow x \]
\[ f(i(x), x) \rightarrow e \]
\[ f(x, i(x)) \rightarrow e \]
\[ i(e) \rightarrow e \]
\[ i(i(x)) \rightarrow x \]
\[ i(f(x, y)) \rightarrow f(i(y), i(x)) \]
\[ f(i(x), f(x, y)) \rightarrow y \]
\[ f(x, f(i(x), y)) \rightarrow y \]
\[ f(f(x, y), z) \rightarrow f(x, f(y, z)) \]
Inference Rules of Completion

Inference system works on pairs of a set of equations E and TRS R

**DELETE:** \((E \cup \{s \leftrightarrow s\}; R) \vdash (E; R)\)

**ORIENT:** \((E \cup \{s \leftrightarrow t\}; R) \vdash (E; R \cup \{s \rightarrow t\})\) if \(s \succ t\)

**SIMPLIFY:** \((E \cup \{s \leftrightarrow t\}; R) \vdash (E \cup \{s \leftrightarrow u\}; R)\) if \(t \rightarrow_R u\)

**COMPOSE:** \((E; R \cup \{s \rightarrow t\}) \vdash (E; R \cup \{s \rightarrow u\})\) if \(t \rightarrow_R u\)

**COLLAPSE:** \((E; R \cup \{t \rightarrow s\}) \vdash (E \cup \{u \leftrightarrow s\}; R)\) if \(l \rightarrow r \in R, \ t \rightarrow_{\{l \rightarrow r\}} u, \) and \(t \triangleright l\)

**DEDUCE:** \((E; R) \vdash (E \cup \{s \leftrightarrow t\}; R)\) if \(s \leftarrow_R u \rightarrow_R t\)

---

Start: \((E,\{\})\)

E is given as input

\[
\begin{align*}
E & \quad \begin{array}{l}
E \quad a = b \\
R \quad a = c
\end{array} \\
& \quad \text{ORIENT}
\end{align*}
\]

reduction order: \(a > b > c\)

adding equation to keep confluence

Goal: \((\{\},R)\)

R is convergent

\[
\begin{align*}
E & \quad \begin{array}{l}
R \quad a \rightarrow b \\
a \rightarrow c
\end{array} \\
& \quad \text{ORIENT, SIMPLIFY}
\end{align*}
\]

\[
\begin{align*}
E & \quad \begin{array}{l}
R \quad b \rightarrow c \\
a \rightarrow c
\end{array} \\
& \quad \text{Goal:} \quad (\{\},R)
\end{align*}
\]

orients equations to keep termination
Problems of Completion Procedure

- procedure needs **appropriate reduction order** (well-founded strict partial order on terms) to ensure the termination of generated system
- if reduction order is not appropriate, procedure may fail or **diverge**
  - impossible to try out some orders sequentially

- Improved completion procedure that **searches** for orders
  - **Multi-Completion** [Kurihara and Kondo: 1999]
    - simulates parallel processes for completion procedure
  - **Constraint-based Completion** [Wehrman et al.: 2006]
    - use modern termination checker in orientation
Multi-Completion

- simulates parallel execution of standard completion
  - accept a set of orders \( \{>_1, \ldots, >_n\} \) and simulates \( P_1, \ldots, P_n \)
  - each process \( P_i \) works with an order \( >_i \)

- node \( <s : t, R_1, R_2, E> \): key data structure for efficiency
  - \( s, t \): called datum, ordered pair of terms
  - \( R_1, R_2, E \): called label, set of indexes of the processes
    - \( i \in R_1(R_2) \) implies \( P_i \) has the rule \( s \rightarrow t \) (\( t \rightarrow s \))
    - \( i \in E \) implies \( P_i \) has the equation \( s = t \)

\[
\begin{array}{c}
P_1 \quad P_2 \quad P_3 \\
E \quad a = b \quad E \quad a = b \quad E \quad \\
R \quad \quad R \quad \quad R \quad \quad \\
\quad c \rightarrow b \quad \quad b \rightarrow c \quad a \rightarrow b \\
\end{array}
\]

Node set representation

\[
\begin{array}{c}
\n\langle a:b, \{3\}, \{\}, \{1,2\} \rangle \\
\langle b:c, \{2,3\}, \{1\}, \{\} \rangle \\
\end{array}
\]
Constraint-based Completion

[Wehrman, Stump, Westbrook: 2006]

- uses (automated) **termination checker** for ensuring termination of generated system
- no order is given

- to ensure consistency of (implicit) reduction order, keeps all rules generated before now (**constraint system**)

\[
\text{ORIENT} : \quad (E \cup \{s \leftrightarrow t\}, R, C) \vdash \\
\quad (E, R \cup \{s \rightarrow t\}, C \cup \{s \rightarrow t\})
\]

if \( C \cup \{s \rightarrow t\} \) terminates

- **terminating constraint system** defines reduction order
- **more powerful** termination proving method can be used
- more time-consuming
Searching appropriate direction

- Equations can be oriented in two ways
  - difficult to decide appropriate direction in orientation
- For successful completion, it’s better to try both ways
  - Since the process can diverge, depth-first search is inappropriate

For efficient simulation, same approach as Multi-Completion can be applied
Multi-Completion based on Constraint System

- Keeping constraint system
  - Add new labels $C_1, C_2$ to Node: $<s: t, R_1, R_2, E, C_1, C_2>$
  - $i \in C_1(C_2)$ implies
    process $P_i$ has rule $s \rightarrow t$ ($t \rightarrow s$) in constraint system

- Trying both directions in ORIENT rule
  - Expresses **splitting of processes** by label operation

```
P_1
  a = b
  b = c
  d \rightarrow c

P_1 \Rightarrow P_2, P_3

P_2
  a \rightarrow b
  b = c
  d \rightarrow c

P_3
  a \leftarrow b
  b = c
  d \rightarrow c

\begin{align*}
\langle a:b, \emptyset, \emptyset, \{1\}, \emptyset, \emptyset \rangle \\
\langle b:c, \emptyset, \emptyset, \{1\}, \emptyset, \emptyset \rangle \\
\langle c:d, \emptyset, \{1\}, \emptyset, \emptyset, \{1\} \rangle
\end{align*}

N

\begin{align*}
\langle a:b, \{2\}, \{3\}, \emptyset, \{2\}, \{3\} \rangle \\
\langle b:c, \emptyset, \emptyset, \{2,3\}, \emptyset, \emptyset \rangle \\
\langle c:d, \emptyset, \{2,3\}, \emptyset, \emptyset, \{2,3\} \rangle
\end{align*}

N'
```

```
handles target node for orientation specially

replacing all 1 by 2,3
```
### Experiments 1

Problem: [Steinbach, 1990], [Wehrman, 2006]

<table>
<thead>
<tr>
<th>Problems</th>
<th>naive approach</th>
<th>node-based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>rewriting</td>
</tr>
<tr>
<td>SK90_3.04</td>
<td>190.3</td>
<td>150.1</td>
</tr>
<tr>
<td>SK90_3.06</td>
<td>3.6</td>
<td>2.1</td>
</tr>
<tr>
<td>SK90_3.07</td>
<td>4.1</td>
<td>2.3</td>
</tr>
<tr>
<td>SK90_3.27</td>
<td>21.1</td>
<td>3.2</td>
</tr>
<tr>
<td><strong>SK90_3.28</strong></td>
<td>410.5</td>
<td><strong>133.4</strong></td>
</tr>
<tr>
<td>WSW07_CGE₂</td>
<td>435.9</td>
<td>272.4</td>
</tr>
<tr>
<td><strong>WS06_PR</strong></td>
<td>28074.7</td>
<td><strong>14690.5</strong></td>
</tr>
</tbody>
</table>

- In all problems, time for **rewriting** is **reduced**
- It is **effective** for the problems which requires a long time and run many processes
Problems of new procedure

- New procedure can execute some basic operations which occurs many processes in common all at once
- Basic operations are rewriting, matching and unification
- But **checking termination** get executed for each process independently
- For almost every problems, **more than 90%** of the running time is termination checking

- Needs of efficient termination checking of **some TRSs** (especially, which has some rules in common)

Used only **Dependency-pair method (DP)** for termination proving and **efficiently convert** DP-problem to SAT
Efficient Termination Checking with DP

From TRS R, calculate a set of “term” constraints $s \geq (> t)$ on reduction order based on DP

- For each term constraint, convert $s \geq (> t)$ to diophantine constraint and propositional formula
- Convert conjunction of all formulas to CNF, check satisfiability by SAT solver

Technique (1) calculates common constraints in some processes from node structure

Technique (2) caches pairs of term constraint and propositional formula
Experiments 2

Problem: [Steinbach, 1990], [Wehrman, 2006]

<table>
<thead>
<tr>
<th>Problem</th>
<th>no techniques</th>
<th>technique (1)</th>
<th>technique (1) and (2)</th>
<th>rate of reduced time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK90_3.04</td>
<td>63.9</td>
<td>60.7</td>
<td>55.7</td>
<td>12.8</td>
</tr>
<tr>
<td>SK90_3.06</td>
<td>2.1</td>
<td>2.1</td>
<td>1.9</td>
<td>9.5</td>
</tr>
<tr>
<td>SK90_3.07</td>
<td>2.5</td>
<td>2.4</td>
<td>2.3</td>
<td>8.0</td>
</tr>
<tr>
<td>SK90_3.27</td>
<td>20.9</td>
<td>20.1</td>
<td>19.1</td>
<td>8.6</td>
</tr>
<tr>
<td>SK90_3.28</td>
<td>353.5</td>
<td>253.3</td>
<td>207.9</td>
<td>41.2</td>
</tr>
<tr>
<td>WSW07_CGE2</td>
<td>199.13</td>
<td>157.7</td>
<td>126.4</td>
<td>36.5</td>
</tr>
<tr>
<td>WSW07_CGE3</td>
<td>53885.4</td>
<td>41256.0</td>
<td>32867.6</td>
<td>39.0</td>
</tr>
<tr>
<td>WS06_PR</td>
<td>16041.1</td>
<td>13210.8</td>
<td>10752.1</td>
<td>32.9</td>
</tr>
</tbody>
</table>

For **difficult problems**, these techniques are *effective*
Conclusion and Future Work

- **Conclusion**
  - we have proposed new completion procedure which efficiently simulates parallel execution of constraint-based completion
  - Experiments show that
    - New procedure is more efficient than naive approach
    - It is effective applying our techniques combined with dependency-pair method

- **Future work**
  - introduce extensions for standard completion
    - AC-completion, unfailing completion
  - application to formal verification
    - inductive theorem proving, E-unification