Design and Implementation of Multi-Context Rewriting Induction

Haruhiko Sato and Masahito Kurihara
Hokkaido University, Japan
Outline

• Inductive Theorem Proving
• Rewriting Induction
• Strategic Difficulties in Rewriting Induction
  • Examples
• Multi-Context Rewriting Induction
• Experimental Results
Inductive theorem proving

**Mathematical Induction Principle**

\[ P(0) \land [P(x) \Rightarrow P(x + 1)] \Rightarrow \forall x \in \mathbb{N}. P(x) \]

**Well-founded Induction Principle**

\[ \succ : \text{well-founded order on } A \]

\[ [x \succ y \Rightarrow P(y)] \Rightarrow P(x) \Rightarrow \forall x \in A. P(x) \]

various properties of programs can be proved by the induction principle
Inductive theorems in Programs

Program: Term Rewriting System

\[ \mathcal{R} = \begin{cases} 
+ (0, y) & \rightarrow y \\
+ (s(x), y) & \rightarrow s(+ (x, y)) \\
\text{len}([]) & \rightarrow 0 \\
\text{len}(x : xs) & \rightarrow s(\text{len}(xs)) \\
@([], ys) & \rightarrow ys \\
@ (x : xs, ys) & \rightarrow x : @ (xs, ys) 
\end{cases} \]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s(0)</td>
<td>s(s(0))</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[]</th>
<th>[x]</th>
<th>[x, y]</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[]</td>
<td>x:[]</td>
<td>x : (y : [])</td>
<td>...</td>
</tr>
</tbody>
</table>

Execution of Program: Rewriting (application of rules)

\[ + (s(s(0)), 0) \rightarrow_{\mathcal{R}} s(+ (s(0), 0)) \rightarrow_{\mathcal{R}} s(s(+ (0, 0))) \rightarrow_{\mathcal{R}} s(s(0)) \]

Property of Program: Inductive Theorem (prop. which holds for any ground-substitution)

\[ \text{len}(@ (xs, ys)) = + (\text{len}(xs), \text{len}(ys)) \]

resultant term contains no variable
Rewriting Induction [Reddy90, Aoto07]

- Induction principle for programs satisfying termination
- well-founded induction based on the order: \( s \succ t \iff s \rightarrow^+ t \)

Inference Rules of Rewriting Induction

**DELETE**: \( \langle \mathcal{E} \cup \{s = s\}, \mathcal{H} \rangle \vdash \langle \mathcal{E}, \mathcal{H} \rangle \)

**SIMPILIFY**: \( \langle \mathcal{E} \cup \{s = t\}, \mathcal{H} \rangle \vdash \langle \mathcal{E} \cup \{s = t'\}, \mathcal{H} \rangle \) if \( t \rightarrow_{\mathcal{R} \cup \mathcal{H}} t' \)

**EXPAND**: \( \langle \mathcal{E} \cup \{s = t\}, \mathcal{H} \rangle \vdash \langle \mathcal{E} \cup \text{Expd}_u(s, t), \mathcal{H} \cup \{s \rightarrow t\} \rangle \)

if \( u \in \mathcal{B}(s), \mathcal{R} \cup \mathcal{H} \cup \{s \rightarrow t\} \) is terminating

**Theorem**

\( \langle \mathcal{E}, \{\} \rangle \vdash \cdots \vdash \langle \{\}, \mathcal{H} \rangle \Rightarrow \mathcal{E} \) is inductive theorem of \( \mathcal{R} \)
Example: associativity of addition

\[ \mathcal{R} = \begin{cases} 
+ (0, y) & \rightarrow y \\
+ (s(x), y) & \rightarrow s(+ (x, y)) 
\end{cases} \]

\[ \mathcal{E} = \begin{cases} 
+ (+ (x, y), z) & = + (x, + (y, z)) 
\end{cases} \]

\[ \left\langle \{ + (+ (x, y), z) = + (x, + (y, z)) \} , \{ \} \right\rangle \]

EXPAND
choose a basic term to expand

\[ + (+ (x, y), z) \]

SIMPLIFY
determine substitutions by unification with rules
\[ \sigma = \{ x \mapsto 0 \} , \{ x \mapsto s(x) \} \]

\[ (+ + (0, y), z) \]

\[ (+ + (s(x), y), z) \]

reduce the instanciated terms

\[ + (y, z) \]

\[ + (s(+ (x, y)), z) \]

DELETE

\[ \left\langle \{ \} , \{ + (+ (x, y), z) \rightarrow + (x, + (y, z)) \} \right\rangle \]
Problem: How do we apply the inference rules?

Standard Strategy

1. apply SIMPLIFY as much as possible
2. apply DELETE as much as possible
3. apply EXPAND to a conjecture
4. go back to 1

(1) which rewrite rule to be chosen?
(2) which direction the conjecture to be oriented for?
(3) which basic subterm to be expanded?

the way of applying inference rules determines contexts: conjectures to be proved, inductive hypotheses

choosing appropriate contexts is important to obtain successful proofs
Example: appropriate **simplification** is required

\[ R = \begin{cases} \emptyset \rightarrow ys \\ (x : xs) \rightarrow x : (xs@ys) \\ r(\emptyset) \rightarrow \emptyset \\ r(x : xs) \rightarrow r(xs)@[x] \end{cases} \]

\[ E = \begin{cases} r(xs) = b(xs) \\ b(b(xs)) = xs \end{cases} \]

**EXPAND**

\[ \langle \{ \begin{array}{c} r(xs) = b(xs) \\ b(b(xs)) = xs \end{array} \} \rangle \]

**EXPAND**

\[ \langle \{ \begin{array}{c} b(xs)@[x] = b1(x, xs) : b2(x, xs) \\ b(b1(x, xs) : b2(x, xs)) = x : xs \end{array} \} \rangle \]

**simplify** → **proof fails** (diverge)

**simplify** → **proof succeeds**
Example: appropriate **orientation** is required

\[ \mathcal{R} = \begin{cases} 
  f(0) & \rightarrow & 0 \\
  f(s(0)) & \rightarrow & s(0) \\
  f(s(s(x))) & \rightarrow & f(s(x)) + f(x) \\
  g(0) & \rightarrow & \langle s(0), 0 \rangle \\
  g(s(x)) & \rightarrow & \text{np}(g(x)) \\
  \text{np}(\langle x, y \rangle) & \rightarrow & \langle x+y, x \rangle 
\end{cases} \]

\[ \mathcal{E} = \{ \langle f(s(x)), f(x) \rangle = g(x) \} \]

orient from left to right: **diverge**

\[ \langle f(s(x)), f(x) \rangle = g(x) \]
\[ \langle f(s(x)) + f(x), f(s(x)) \rangle = \text{np}(g(x)) \]
\[ \langle (f(s(x)) + f(x)) + f(s(x)), f(s(x)) + f(x) \rangle = \text{np}(\text{np}(g(x))) \]

orient from right to left: **succeeds**

\[ \langle f(s(s(x))), f(s(x)) \rangle = \text{np}(g(x)) \rightarrow_{\mathcal{H}} \text{np}(\langle f(s(x)), f(x) \rangle) \]
\[ \rightarrow_{\mathcal{R}} \langle f(s(x)) + f(x), f(s(x)) \rangle \rightarrow_{\mathcal{R}} \langle f(s(x)) + f(x), f(s(x)) \rangle \]
Example: appropriate **expansion** is required

\[ \mathcal{R} = \left\{ \begin{array}{ll}
\text{or}(x, \text{true}) & \rightarrow \text{true} \\
\text{or}(x, \text{false}) & \rightarrow x \\
\text{and}(x, \text{true}) & \rightarrow x \\
\text{and}(x, \text{false}) & \rightarrow \text{false} \\
\text{le}(0, x) & \rightarrow \text{true} \\
\text{le}(s(x), 0) & \rightarrow \text{false} \\
\text{le}(s(x), s(y)) & \rightarrow \text{le}(x, y)
\end{array} \right. \]

\[ \mathcal{E} = \{ \text{imply}(\text{and}(\text{eq}(x, y), \text{eq}(y, z)), \text{eq}(x, z)) \} = \text{true} \]

Expansion with a specific (leftmost-outermost) strategy causes divergence

\[
\begin{align*}
(x = y) \land (y = z) & \implies (x = z) \\
(x = y) \land (y + 1 = z) & \implies (x + 1 = z) \\
(x = y) \land (y + 2 = z) & \implies (x + 2 = z) \\
\vdots & \quad \vdots \quad \vdots
\end{align*}
\]

The divergence can be avoided by appropriate choice of expansion position.
Our Approach: Trying multiple contexts in parallel

<table>
<thead>
<tr>
<th></th>
<th>Traditional Approach</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simplification</strong></td>
<td>normalize w.r.t. a specific reduction strategy</td>
<td>calculate all of normal forms</td>
</tr>
<tr>
<td><strong>Orientation</strong></td>
<td>try a direction (w.r.t. the order given by user)</td>
<td>try both direction</td>
</tr>
<tr>
<td><strong>Expansion</strong></td>
<td>try one of basic subterms</td>
<td>try all of basic subterms</td>
</tr>
</tbody>
</table>
Example: trying multiple contexts in orientation

- A process \( p \) encounters \( n \) nondeterministic choices.
- Fork the process \( p \) into \( n \) different processes, and assign them with the choices.

Example: orientation

Consider a process \( p \) with the following rules:

\[
\begin{align*}
\langle \{ & \text{r}(xs) = \text{b}(xs) \}, \{ \} \rangle \\
\{ & \text{b}(\text{b}(xs)) = xs \}
\end{align*}
\]

Orient from left to right:

- Process \( p1 \):
  \[
  \langle \{ \text{r}(xs)@[x] = \text{b}(x:xs) \}, \{ \text{b}(\text{b}(xs)) = xs \}, \{ \text{r}(xs) \rightarrow \text{b}(xs) \} \rangle
  \]

Orient from right to left:

- Process \( p2 \):
  \[
  \langle \{ \text{r}(x:xs) = \text{b1}(x,xs) : \text{b2}(x,xs) \}, \{ \text{b}(\text{b}(xs)) = xs \}, \{ \text{b}(xs) \rightarrow \text{r}(xs) \} \rangle
  \]

This example demonstrates how nondeterministic choices can be handled in process orientation.
Representation of Multiple Processes

use node-representation in multi-completion [Kurihara et al. 1999, Sato et al. 2008]

process $p_1$

\[
\begin{align*}
  \{ r(xs)@[x] = b(x : xs) \} \\
  \{ b(b(xs)) = xs \} \\
  \{ r(xs) \rightarrow b(xs) \}
\end{align*}
\]

process $p_2$

\[
\begin{align*}
  \{ r(x : xs) = b_1(x, xs) : b_2(x, xs) \} \\
  \{ b(b(xs)) = xs \} \\
  \{ b(xs) \rightarrow r(xs) \}
\end{align*}
\]

representation by a set of nodes

\[\langle s, t, H_1, H_2, E \rangle\]

ordered pair of terms $s, t$

set of processes which keeps:

\[
\begin{align*}
  s \rightarrow t & \quad t \rightarrow s & \quad s = t
\end{align*}
\]

\[
\begin{align*}
  \{ r(xs)@[x], b(x : xs), \} \\
  \{ b(b(xs)), xs, \} \\
  \{ r(xs), b(xs), \} \\
  \{ r(x : xs), b_1(x, xs) : b_2(x, xs) \}
\end{align*}
\]

\[
\begin{align*}
  \{ \}, \quad \{ \}, \quad \{ p_1 \} \ \\
  \{ \}, \quad \{ \}, \quad \{ p_1, p_2 \} \ \\
  \{ p_1 \}, \quad \{ p_2 \}, \quad \{ \} \ \\
  \{ \}, \quad \{ \}, \quad \{ p_2 \}
\end{align*}
\]

- can share common inferences among processes
- efficient to fork/remove processes \{\ldots, p, \ldots\} \rightarrow \{\ldots, p_1, p_2, \ldots, p_n, \ldots\}
- low memory consumption
Multi-Context Rewriting Induction (MRI)

(a part of) Inference Rules of MRI

defined on a set of nodes, and simulates a lot of inferences in a single operation

\textbf{Simplify-R:} \quad N \cup \{ \langle s : t, H_1, H_2, E \rangle \} \vdash
\begin{align*}
N \cup & \{ \langle s : t, H_1, H_2, \emptyset \rangle \} \\
N \cup & \{ \langle s' : t, \emptyset, \emptyset, E \rangle \} \\
\text{if } & E \neq \emptyset \text{ and } s \rightarrow_{R} s'
\end{align*}

processes keeps \( s = t \)

\begin{align*}
\textbf{Simplify-H:} \quad N \cup & \{ \langle s : t, H_1, H_2, E \rangle \} \vdash \\
N \cup & \{ \langle s : t, H_1, H_2, E \setminus H \rangle \} \\
N \cup & \{ \langle s' : t, \emptyset, \emptyset, E \cap H \rangle \} \\
\text{if } & E \cap H \neq \emptyset, \langle l : r, H, \ldots, \ldots \rangle \in N, \\
\text{and } & s \rightarrow_{\{l \rightarrow r\}} s'
\end{align*}

processes which can infer \( s \rightarrow_{\{l \rightarrow r\}} s' \)

processes keeps \( l \rightarrow r \)
Experimentation

environment: Intel Xeon 2.13GHz, 1GB system memory

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time (sec)</th>
<th># of processes</th>
<th>Estimated Time (sec)</th>
<th>Reduced Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>list reverse</td>
<td>0.23</td>
<td>10</td>
<td>0.24</td>
<td>0.01</td>
</tr>
<tr>
<td>fibonacci</td>
<td>0.01</td>
<td>3</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>transitivity of equality</td>
<td>1.37</td>
<td>127</td>
<td>4.32</td>
<td>2.95</td>
</tr>
</tbody>
</table>

our node-based approach is **effective for difficult problems** which requires large number of processes
Conclusion and Future work

**Conclusion**

- We have shown some examples in which the appropriate choice of contexts is important to obtain successful results in RI.
- We have proposed a new inference system which proves inductive theorems efficiently and automatically.
- Experimentally, we have shown that our system can prove some difficult problems in reasonable time.

**Future Work**

- Strategies for giving precedence to important conjectures.
- More efficient termination checking for incremental systems.