Rewriting Tools

Nao Hirokawa

JAIST

14th ISR, August 26, 2024

`https://www.jaist.ac.jp/~hirokawa/24isr/`

Schedule

1. Monday: rewriting tools
2. Tuesday: termination tools
3. Wednesday: confluence tools
4. Friday: completion tools

Keywords

SAT/SMT, unification, tree automata, redundant rules

Rewriting Tools: CafeOBJ, Maude, K, ...

- based on order-sorted conditional AC-rewriting
- support reachability analysis
- used for software verification (rewriting logic)

Example of CafeOBJ Code

```
open EQL .
[N]
op   0 : -> N .
op   s : N -> N .
op   _+_ : N N -> N {assoc comm}.
vars X Y : N .
eq   0 + Y    = Y .
eq   s(X) + Y = s(X + Y) .
red  (X + 0) + s(X) .   -- reduces to s(X + X)
close .
```

Contents

Innermost Rewriting

Innermost Rewriting

Definition

$s \xrightarrow{\text{i}}_{\mathcal{R}} t \iff \exists \ell \to r \in \mathcal{R}, C, \sigma.\ \begin{cases} s = C[\ell\sigma],\ t = C[r\sigma], \text{ and} \\ u \in \mathsf{NF}(\mathcal{R}) \text{ for all proper subterms } u \text{ of } \ell\sigma \end{cases}$

Example

TRS

$$0 + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

- $0 + \mathsf{s}(\underline{\mathsf{s}(0) + 0}) \xrightarrow{\text{i}}_{\mathcal{R}} 0 + \mathsf{s}(\mathsf{s}(0 + 0))$
- $\underline{0 + \mathsf{s}(\mathsf{s}(0) + 0)} \not\xrightarrow{\text{i}}_{\mathcal{R}} \mathsf{s}(\mathsf{s}(0) + 0)$

Innermost Normalization – Naive Version

Definition

given TRS $\mathcal{R}$, function $\phi(t)$ is defined as follows:

$$\phi(x) = x$$

$$\phi(f(t_1, \dots, t_n)) = \begin{cases} \phi(r\tau) & \text{if } \ell \to r \in \mathcal{R} \text{ and } t' = \ell\tau \\ \underbrace{t'}_{\mathsf{NF}} & \text{otherwise} \end{cases}$$

where $t' = f(\underbrace{\phi(t_1)}_{\mathsf{NF}}, \dots, \underbrace{\phi(t_n)}_{\mathsf{NF}})$

Theorem

if t is innermost terminating then $\phi(t)$ is normal form of t

Example of Innermost Normalization

TRS

$$0 + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

innermost normalization:

$$\underbrace{\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}}_{\text{matched}} \xrightarrow{\mathsf{i}} \mathsf{s}(\underbrace{\underbrace{0}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}}_{\text{matched}}) \xrightarrow{\mathsf{i}} \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}$$

Observation

substitutions employed in innermost steps are normalized

Exploiting Normalized Substitutions

TRS

$$0 + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

innermost normalization:

$$\underbrace{\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{\mathsf{s}(\underbrace{0}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}})}_{\mathsf{NF}}}_{\text{matched}} = (\mathsf{s}(x) + y)\underbrace{\sigma}_{\mathsf{NF}} \xrightarrow{\mathsf{i}} \mathsf{s}(x + y)\underbrace{\sigma}_{\mathsf{NF}} = \mathsf{s}(\underbrace{\underbrace{0}_{\mathsf{NF}} + \underbrace{\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))}_{\mathsf{NF}}}_{\text{matched}}) \xrightarrow{\mathsf{i}} \cdots$$

where $\sigma = \begin{Bmatrix} x \mapsto 0 \\ y \mapsto \mathsf{s}(\mathsf{s}(\mathsf{s}(x))) \end{Bmatrix}$

Innermost Normalization – Efficient Version

Definition

given TRS $\mathcal{R}$, operator $t * \sigma$ is defined as follows:

$$x * \sigma = x\sigma$$

$$f(t_1, \ldots, t_n) * \sigma = \begin{cases} r * \underbrace{\tau}_{\mathsf{NF}} & \text{if } \ell \to r \in \mathcal{R} \text{ and } t' = \ell \underbrace{\tau}_{\mathsf{NF}} \\ \underbrace{t'\sigma}_{\mathsf{NF}} & \text{otherwise} \end{cases}$$

where $t' = f(\underbrace{t_1 * \sigma}_{\mathsf{NF}}, \ldots, \underbrace{t_n * \sigma}_{\mathsf{NF}})$

Theorem (folklore?)

if t is innermost terminating and σ is normalized substitution then

$t * \sigma$ is normal form of $t\sigma$

AC Rewriting

AC Rewriting

let $\mathcal{F}_{\mathsf{AC}}$ be set of binary symbols (AC symbols)

Definition

$$\mathsf{AC} = \left\{ \begin{array}{c} f(x,y) \to f(y,x) \\ f(f(x,y),z) \to f(x,f(y,z)) \end{array} \;\middle|\; f \in \mathcal{F}_{\mathsf{AC}} \right\}$$

Example

$(\mathsf{a}+\mathsf{b})+\mathsf{c} \leftrightarrow^*_{\mathsf{AC}} (\mathsf{b}+\mathsf{a})+\mathsf{c} \leftrightarrow^*_{\mathsf{AC}} \mathsf{c}+(\mathsf{b}+\mathsf{a})$ if $+$ is AC symbol

Definition (class rewriting)

$s \to_{\mathcal{R}/\mathsf{AC}} t$ if $s \leftrightarrow^*_{\mathsf{AC}} \cdot \to_{\mathcal{R}} \cdot \leftrightarrow^*_{\mathsf{AC}} t$

Example of AC Rewriting

TRS $\mathcal{R}$ with AC symbol $+$:

$$\mathcal{R} = \left\{ \begin{array}{c} 0+x \to x \\ \mathsf{s}(x)+y \to \mathsf{s}(x+y) \\ \infty+\infty \to \infty \end{array} \right\} \qquad \mathsf{AC} = \left\{ \begin{array}{c} (x+y)+z \to x+(y+z) \\ x+y \to y+x \end{array} \right\}$$

AC rewriting

- $(x+0)+\mathsf{s}(x) \to_{\mathcal{R}/\mathsf{AC}} x+\mathsf{s}(x) \to_{\mathcal{R}/\mathsf{AC}} \mathsf{s}(x+x) \in \mathsf{NF}(\to_{\mathcal{R}/\mathsf{AC}})$
- $\infty+(x+\infty) \to_{\mathcal{R}/\mathsf{AC}} \infty+x \in \mathsf{NF}(\to_{\mathcal{R}/AC})$

how to implement $\to_{\mathcal{R}/\mathsf{AC}}$? ☞ rewriting based on AC matching

Rewriting based on AC Matching

Definition (Peterson and Stickel 1981)

$s \to_{\mathcal{R},\mathsf{AC}} t$ if $s|_p \leftrightarrow^*_{\mathsf{AC}} \ell\sigma$ and $t = s[r\sigma]_p$ for some $p \in \mathcal{P}\mathsf{os}(s)$ and $\ell \to r \in \mathcal{R}$

Example

$$\mathcal{R} = \left\{ \begin{array}{c} 0+x \to x \\ \mathsf{s}(x)+y \to \mathsf{s}(x+y) \\ \infty+\infty \to \infty \end{array} \right\} \qquad \mathsf{AC} = \left\{ \begin{array}{c} (x+y)+z \to x+(y+z) \\ x+y \to y+x \end{array} \right\}$$

- $(x+0)+\mathsf{s}(x) \to_{\mathcal{R},\mathsf{AC}} x+\mathsf{s}(x) \to_{\mathcal{R},\mathsf{AC}} \mathsf{s}(x+x) \in \mathsf{NF}(\to_{\mathcal{R},\mathsf{AC}})$
- $\infty+(x+\infty) \not\to_{\mathcal{R},\mathsf{AC}} \infty+x$!?

Coherence Completion

Definition

- $f(\ell,x) \to f(r,x)$ is extension rule of $\ell \to r$ if

  $f = \mathsf{root}(\ell) \in \mathsf{AC}$ and $x \notin \mathcal{V}\mathsf{ar}(\ell)$
- $\mathcal{R}^e$ is extension of $\mathcal{R}$ with extension rules

Theorem

$s \to_{\mathcal{R}/\mathsf{AC}} t \iff s \to_{\mathcal{R}^e,\mathsf{AC}} \cdot \leftrightarrow^*_{\mathsf{AC}} t$

Example of AC Rewriting

$$\mathcal{R}^e = \left\{ \begin{array}{ll} 0 + x \to x & (0+x)+y \to x+y \\ \mathsf{s}(x) + y \to \mathsf{s}(x+y) & (\mathsf{s}(x)+y)+z \to \mathsf{s}(x+y)+z \\ \infty + \infty \to \infty & (\infty+\infty)+x \to \infty + x \end{array} \right\}$$

$\infty + (\infty + x) \to_{\mathcal{R}^e,\mathsf{AC}} \infty + x$ because

$$\infty + (\infty + x) \leftrightarrow_{\mathsf{AC}} (\infty + \infty) + x \xrightarrow{\epsilon}_{\mathcal{R}^e} \infty + x$$

AC Matching

AC Matching Problem

Definition

AC matching problem is following problem:

input: terms s and t — s is called pattern

output: substitution σ with $s\sigma \leftrightarrow^*_{\mathsf{AC}} t$ if it exists

Example

let $+$ be AC symbol

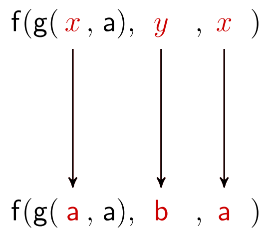
$$(\mathsf{a} + x + \mathsf{b})\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{b} + \mathsf{s}(y) + \mathsf{a} \quad \text{if } \sigma = \{x \mapsto \mathsf{s}(y)\}$$

$$(x + y)\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{b} \quad \text{if } \sigma = \left\{\begin{array}{l} x \mapsto \mathsf{a} \\ y \mapsto \mathsf{b}\end{array}\right\} \text{ or } \sigma = \left\{\begin{array}{l} x \mapsto \mathsf{b} \\ y \mapsto \mathsf{a}\end{array}\right\}$$

Exercises: Solve AC Matching Problems

1. $(x + \mathsf{a})\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{b} + \mathsf{b}$ — $\{x \mapsto \mathsf{b} + \mathsf{b}\}$
2. $(x + \mathsf{a})\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{a}$ — $\{x \mapsto \mathsf{a}\}$
3. $(x + x)\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{b}$ — no solution
4. $(x + y + z)\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{b}$ — no solution
5. $(x + x)\sigma \leftrightarrow^*_{\mathsf{AC}} \mathsf{a} + \mathsf{a} + \mathsf{b} + \mathsf{b}$ — $\{x \mapsto \mathsf{a} + \mathsf{b}\}$, $\{x \mapsto \mathsf{b} + \mathsf{a}\}$

Ordinary Matching Algorithm

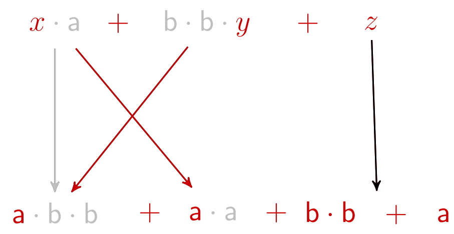


$$\left\{\begin{array}{l} x \mapsto \mathsf{a} \\ y \mapsto \mathsf{b} \end{array}\right\}$$

Theorem (folklore)

ordinary matching problem is solvable in polynomial time

AC Matching Algorithm for Linear Patterns



$$\left\{\begin{array}{l} x \mapsto \mathsf{a} \\ y \mapsto \mathsf{a} \\ z \mapsto \mathsf{b} \cdot \mathsf{b} + \mathsf{a} \end{array}\right\}$$

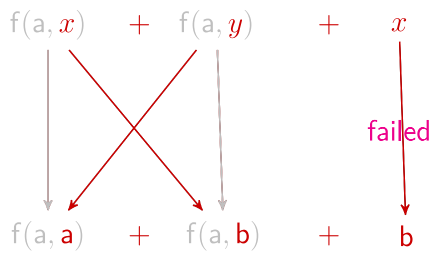
Theorem (Benanav et al. 1987)

AC matching problem for linear patterns is solvable in polynomial time

Proof.

use polynomial time algorithm for bipartite matching (Hopcroft and Karp 1973) □

AC Matching Algorithm for General Patterns



$$\left\{\begin{array}{l} x \mapsto \mathsf{b} \\ y \mapsto \mathsf{a} \end{array}\right\}$$

Note

backtracking is necessary; efficient pruning technique is known (Eker 1995)

Note

AC matching is NP-complete; how to prove it? — bin packing problem

Bin Packing is NP-Complete

Theorem

following bin packing problem is NP-complete:

instance: *multiset M of positive numbers and $n, B \in \mathbb{N}$*

question: *is there partition $M = M_1 \uplus \cdots \uplus M_n$ with $\sum M_i \leqslant B$*

Example

let $M = \{2, 2, 2, 4, 5, 6, 6\}$

- if $B = 10$ and $n = 3$ then $M = \{2, 2, 6\} \uplus \{2, 5\} \uplus \{4, 6\}$
- if $B = 9$ and $n = 3$ then suitable partition does not exist
- what if $B = 14$ and $n = 2$?

AC Matching is NP-Complete

Theorem (Benanav et al. 1987)

AC matching is NP-complete

Proof (Chandra and Kanellakis 1985).

reduction from bin packing; consider $\{2,2,2,4,5,6,6\}$ with 3 bins of size 10

1. $(x_1^2 \cdot x_2^2 \cdot x_3^2 \cdot x_4^4 \cdot x_5^5 \cdot x_6^6 \cdot x_7^6 \cdot y_{28} \cdot y_{29} \cdot y_{30})\sigma = \mathsf{a}^{10} \cdot \mathsf{b}^{10} \cdot \mathsf{c}^{10}$ has solution:

$$\sigma = \left\{ \begin{array}{r} x_1, x_2, x_6 \mapsto \mathsf{a} \\ x_3, x_5, y_{28}, y_{29}, y_{30} \mapsto \mathsf{b} \\ x_4, x_7 \mapsto \mathsf{c} \end{array} \right\} \qquad \begin{array}{r} \mathsf{a}^2 \cdot \mathsf{a}^2 \cdot \mathsf{a}^6 = \mathsf{a}^{10} \\ \mathsf{b}^2 \cdot \mathsf{b}^5 = \mathsf{b}^7 \\ \mathsf{c}^4 \cdot \mathsf{c}^6 = \mathsf{c}^{10} \end{array}$$

2. $\{2,2,6\} \uplus \{2,5\} \uplus \{4,6\} = \{2,2,2,4,5,6,6\}$ □

Demo

Example: Proof by AC Rewriting

TRS $\mathcal{R}$ with AC symbols $+$ and $\cdot$

$$\begin{array}{ccc} 0 + x \to x & 0 \cdot x \to 0 & \mathsf{f}(0) \to 0 \\ \mathsf{s}(x) + y \to \mathsf{s}(x+y) & \mathsf{s}(x) \cdot y \to x \cdot y + y & \mathsf{f}(\mathsf{s}(x)) \to \mathsf{s}(x) + \mathsf{f}(x) \end{array}$$

$\mathsf{f}(n) + \mathsf{f}(n) \leftrightarrow_{\mathcal{R}}^* n \cdot \mathsf{s}(n)$ with Peano numbers n is shown as follows:

1. use fresh constant c to define extension $\mathcal{S}$ of $\mathcal{R}$ with

$$\mathsf{eq}(x,x) \to \mathsf{true} \quad \mathsf{claim}(x) \to \mathsf{eq}(\mathsf{f}(x) + \mathsf{f}(x), x \cdot \mathsf{s}(x)) \quad \mathsf{f}(\mathsf{c}) + \mathsf{f}(\mathsf{c}) \to \mathsf{n} \cdot \mathsf{s}(\mathsf{c})$$

2. $\mathsf{claim}(0) \to_{\mathcal{S}/\mathsf{AC}}^* \mathsf{true}$
3. $\mathsf{claim}(\mathsf{s}(\mathsf{c})) \to_{\mathcal{S}/\mathsf{AC}}^* \mathsf{true}$

Boolean Ring

```
open EQL .
[B]
op tt : -> B .
op ff : -> B .
op xor : B B -> B {assoc comm} .
op and : B B -> B {assoc comm} .
op or  : B B -> B {assoc comm} .
op imply : B B -> B .
op equiv : B B -> B .
op not : B -> B .
vars x y z : B .
```

```
eq and(x,x)  = x .
eq and(x,ff) = ff .
eq and(x,tt) = x .
eq xor(x,x)  = ff .
eq xor(x,ff) = x .
eq and(x,xor(y,z)) = xor(and(x,y),and(x,z)) .
eq not(x) = xor(x,tt) .
eq or(x,y) = not(and(not(x),not(y))) .
eq imply(x,y) = or(not(x),y) .
eq equiv(x,y) = and(imply(x,y),imply(y,x)) .
red and(tt,or(ff,tt)) .
ops p q r : -> B .
red equiv(imply(p, imply(q, r)),
          imply(and(p, q), r)) .
close .
```

Summary

1. innermost rewriting
2. AC rewriting
3. AC matching

thanks for your attention!