Optimizing mkbTT

S. Winkler H. Sato A. Middeldorp M. Kurihara

Master Seminar 1
Computational Logic Group

January 13, 2010

mkbTT on a shoestring

- mkbTT is completion procedure

$\mathcal{E}$ equations	$+$	$\succ$ reduction ordering	$\xrightarrow{\text{kb}}$	$\mathcal{R}$ rewrite system

$\mathcal{R}$ is confluent, terminating, reduced and $\approx_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

mkbTT on a shoestring

- mkbTT is completion procedure

$$\underset{\text{equations}}{\mathcal{E}} \quad \underset{\substack{\text{mkbTT} \\ +\text{termination prover}}}{\longrightarrow} \quad \underset{\text{rewrite system}}{\mathcal{R}}$$

$\mathcal{R}$ is confluent, terminating, reduced and $\approx_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

mkbTT on a shoestring

- mkbTT is completion procedure

$$\underset{\text{equations}}{\mathcal{E}} \quad \xrightarrow[\text{+termination prover}]{\text{mkbTT}} \quad \underset{\text{rewrite system}}{\mathcal{R}}$$

  $\mathcal{R}$ is confluent, terminating, reduced and $\approx_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

- mkbTT simulates multiple processes
  $P = \{\epsilon, 0, 1, 00, 01, 011, \ldots\}$

mkbTT on a shoestring

- mkbTT is completion procedure

$$\underset{\text{equations}}{\mathcal{E}} \quad \underset{\substack{\text{mkbTT} \\ +\text{termination prover}}}{\longrightarrow} \quad \underset{\text{rewrite system}}{\mathcal{R}}$$

  $\mathcal{R}$ is confluent, terminating, reduced and $\approx_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

- mkbTT simulates multiple processes
  $P = \{\epsilon, 0, 1, 00, 01, 011, \ldots\}$

- mkbTT implements inference system on set of nodes

Definition

node is tuple $\langle s : t, R_0, R_1, E, C_0, C_1 \rangle$ of

- terms s, t
- sets of processes $R_0,\ R_1,\ E,\ C_0,\ C_1$

mkbTT on a shoestring

- mkbTT is completion procedure

$$\underset{\text{equations}}{\mathcal{E}} \quad \underset{\text{+termination prover}}{\overset{\longrightarrow}{\text{mkbTT}}} \quad \underset{\text{rewrite system}}{\mathcal{R}}$$

  $\mathcal{R}$ is confluent, terminating, reduced and $\approx_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

- mkbTT simulates multiple processes
  $P = \{\epsilon, 0, 1, 00, 01, 011, \ldots\}$
- mkbTT implements inference system on set of nodes

Definition

node is tuple $\langle s : t, R_0, R_1, E, C_0, C_1 \rangle$ of

- terms s, t — data
- sets of processes R_0, R_1, E, C_0, C_1 — labels

Outline

- Indexing Techniques
- Selection Strategies
- Critical Pair Criteria
- Isomorphisms
- Conclusion

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

rewrite$_1$
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{N \cup \{\langle s : t, R_1 \setminus R, R_2, E \setminus R, C_1, C_2\rangle\} \cup \{\langle s : u, R_1 \cap R, \varnothing, E \cap R, \varnothing, \varnothing\rangle\}}$$
if $\langle l : r, R, \ldots\rangle \in N$, $t \to_{l\to r} u$, $t \doteq l$

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

rewrite$_2$

$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{\begin{array}{l} N \;\; \cup \{\langle s : t, R_1 \setminus R, R_2 \setminus R, E \setminus R, C_1, C_2\rangle\} \\ \quad\;\; \cup \{\langle s : u, R_1 \cap R, \varnothing, (R_2 \cup E) \cap R, \varnothing, \varnothing\rangle\}\end{array}}$$

if $\langle l : r, R, \ldots\rangle \in N$, $t \rightarrow_{l \rightarrow r} u$, $t \rhd l$

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

deduce $\dfrac{\mathcal{N}}{\mathcal{N} \cup \{\langle s : t, \varnothing, \varnothing, R \cap R', \varnothing, \varnothing \rangle\}}$

if $\langle l : r, R, \ldots \rangle, \langle l' : r', R', \ldots \rangle \in \mathcal{N}$
and $s \leftarrow_{l \to r} u \rightarrow_{l' \to r'} t$

The Term Indexing Problem

Given

- a set of terms L
- a binary relation R on terms
- a term t

identify all $s \in L$ with $s\ R\ t$

The Term Indexing Problem

Given

- a set of terms L — index
- a binary relation R on terms — retrieval condition
- a term t — query term

identify all $s \in L$ with $s\ R\ t$ — candidate terms

The Term Indexing Problem

Given

- a set of terms L — index
- a binary relation R on terms — retrieval condition
- a term t — query term

identify all $s \in L$ with $s\ R\ t$ — candidate terms

Example

mkbTT requires

- variant retrieval in rewrite_1
- encompassment retrieval in rewrite_2
- retrieval of unifiable terms in deduce

The Term Indexing Problem

Given

- a set of terms L — index
- a binary relation R on terms — retrieval condition
- a term t — query term

identify all $s \in L$ with $s\ R\ t$ — candidate terms

Example

mkbTT requires

- variant retrieval in rewrite_1
- encompassment retrieval in rewrite_2
- retrieval of unifiable terms in deduce

Term Indexing Techniques

path indexing, discrimination trees, perfect discrimination trees, adaptive automata, code trees, substitution trees, context trees, ...

The Term Indexing Problem

Given

- a set of terms L — index
- a binary relation R on terms — retrieval condition
- a term t — query term

identify all $s \in L$ with $s \; R \; t$ — candidate terms

Example

mkbTT requires

- variant retrieval in rewrite_1
- encompassment retrieval in rewrite_2
- retrieval of unifiable terms in deduce

Term Indexing Techniques

path indexing, discrimination trees, perfect discrimination trees, adaptive automata, code trees, substitution trees, context trees, ...

Discrimination Trees

Example

- construct index

(1)	$f(g(a,x),c)$	(2)	$f(g(y,b),x)$
(3)	$f(g(a,b),c)$	(4)	$f(g(y,c),b)$
(5)	$f(x,x)$		

Discrimination Trees

Example

- construct index

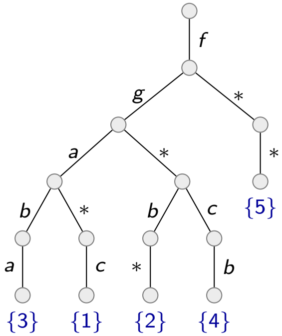
(1)	$f(g(a,x),c)$	f.g.a.*.c	(2)	$f(g(y,b),x)$	f.g.*.b.*
(3)	$f(g(a,b),c)$	f.g.a.b.c	(4)	$f(g(y,c),b)$	f.g.*.c.b
(5)	$f(x,x)$	f.*.*			

Discrimination Trees

Example

- construct index

(1)	$f(g(a,x),c)$	f.g.a.*.c	(2)	$f(g(y,b),x)$	f.g.*.b.*
(3)	$f(g(a,b),c)$	f.g.a.b.c	(4)	$f(g(y,c),b)$	f.g.*.c.b
(5)	$f(x,x)$	f.*.*			



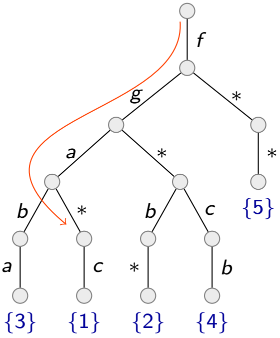
Discrimination Trees

Example

- construct index

(1)	$f(g(a,x),c)$	f.g.a.*.c	(2)	$f(g(y,b),x)$	f.g.*.b.*
(3)	$f(g(a,b),c)$	f.g.a.b.c	(4)	$f(g(y,c),b)$	f.g.*.c.b
(5)	$f(x,x)$	f.*.*			

- find terms matching $f(g(a,c),b))$



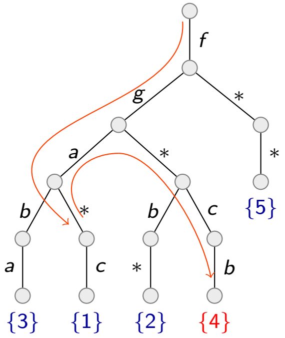
Discrimination Trees

Example

- construct index

(1)	$f(g(a,x),c)$	f.g.a.*.c	(2)	$f(g(y,b),x)$	f.g.*.b.*
(3)	$f(g(a,b),c)$	f.g.a.b.c	(4)	$f(g(y,c),b)$	f.g.*.c.b
(5)	$f(x,x)$	f.*.*			

- find terms matching $f(g(a,c),b))$



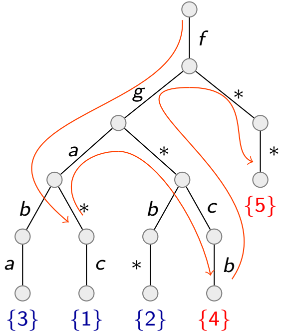
Discrimination Trees

Example

- construct index

(1)	$f(g(a, x), c)$	f.g.a.*.c	(2)	$f(g(y, b), x)$	f.g.*.b.*
(3)	$f(g(a, b), c)$	f.g.a.b.c	(4)	$f(g(y, c), b)$	f.g.*.c.b
(5)	$f(x, x)$	f.*.*			

- find terms matching $f(g(a, c), b))$ $f(g(y, c), b), f(x, x)$



Implementation

- path indexing, discrimination trees, code trees for $\rhd$ and $\doteq$
- path indexing for unifiable terms

Implementation

- path indexing, discrimination trees, code trees for $\rhd$ and $\doteq$
- path indexing for unifiable terms

Experiments

- mkbTT interfacing T$_\mathsf{T}$T$_2$
- 101 systems from various papers, with 600 seconds timeout

Implementation

- path indexing, discrimination trees, code trees for $\rhd$ and $\doteq$
- path indexing for unifiable terms

Experiments

- mkbTT interfacing $\mathsf{T_TT_2}$
- 101 systems from various papers, with 600 seconds timeout

naive			path indexing			discrimination trees			code trees		
(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
1990	387	91	1890	345	18	1650	150	5	1580	106	5

(1) total time for successful completions

Implementation

- path indexing, discrimination trees, code trees for $\rhd$ and $\doteq$
- path indexing for unifiable terms

Experiments

- mkbTT interfacing $\mathsf{T_TT_2}$
- 101 systems from various papers, with 600 seconds timeout

naive			path indexing			discrimination trees			code trees		
(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
1990	387	91	1890	345	18	1650	150	5	1580	106	5

(1) total time for successful completions
(2) total time for $\rhd$ retrieval

Implementation

- path indexing, discrimination trees, code trees for $\rhd$ and $\doteq$
- path indexing for unifiable terms

Experiments

- mkbTT interfacing $\mathsf{T_TT_2}$
- 101 systems from various papers, with 600 seconds timeout

naive			path indexing			discrimination trees			code trees		
(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
1990	387	91	1890	345	18	1650	150	5	1580	106	5

(1) total time for successful completions
(2) total time for $\rhd$ retrieval
(3) total time for $\doteq$ retrieval

Outline

- Indexing Techniques
- Selection Strategies
- Critical Pair Criteria
- Isomorphisms
- Conclusion

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
-

$$
\begin{aligned}
\mathit{strategy} &::= \texttt{?} \mid \texttt{(}\mathit{node\_property}\,\texttt{,}\,\mathit{strategy}\texttt{)} \\
&\quad \mid \mathit{float}\texttt{(}\mathit{strategy}\,\texttt{:}\,\mathit{strategy}\texttt{)} \\
\mathit{node\_property} &::= \texttt{*} \mid \texttt{data(}\mathit{termpair\_property}\texttt{)} \mid \texttt{el(}\mathit{pset\_property}\texttt{)} \\
&\quad \mid \texttt{-}\ \mathit{node\_property} \mid \mathit{node\_property}\ \texttt{+}\ \mathit{node\_property} \\
\mathit{pset\_property} &::= \texttt{\#} \mid \texttt{sum(}\mathit{process\_property}\texttt{)} \mid \texttt{min(}\mathit{process\_property}\texttt{)} \\
\mathit{process\_property} &::= \texttt{e(}\mathit{eqs\_property}\texttt{)} \mid \texttt{r(}\mathit{trs\_property}\texttt{)} \mid \texttt{c(}\mathit{trs\_property}\texttt{)} \\
&\quad \mid \mathit{process\_property}\ \texttt{+}\ \mathit{process\_property} \\
\mathit{trs\_property} &::= \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{cp(}\mathit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{eqs\_property} &::= \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{termpair\_property} &::= \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}
$$

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
-

$$
\begin{aligned}
\mathit{strategy} &::= \; ? \mid (\mathit{node\_property}\,\texttt{,}\,\mathit{strategy}) \\
&\quad\mid \mathit{float}\,\texttt{(}\mathit{strategy}\,\texttt{:}\,\mathit{strategy}\texttt{)} \\
\mathit{node\_property} &::= \; \texttt{*} \mid \texttt{data(}\mathit{termpair\_property}\texttt{)} \mid \texttt{el(}\mathit{pset\_property}\texttt{)} \\
&\quad\mid \texttt{-}\ \mathit{node\_property} \mid \mathit{node\_property}\ \texttt{+}\ \mathit{node\_property} \\
\mathit{pset\_property} &::= \; \texttt{\#} \mid \texttt{sum(}\mathit{process\_property}\texttt{)} \mid \texttt{min(}\mathit{process\_property}\texttt{)} \\
\mathit{process\_property} &::= \; \texttt{e(}\mathit{eqs\_property}\texttt{)} \mid \texttt{r(}\mathit{trs\_property}\texttt{)} \mid \texttt{c(}\mathit{trs\_property}\texttt{)} \\
&\quad\mid \mathit{process\_property}\ \texttt{+}\ \mathit{process\_property} \\
\mathit{trs\_property} &::= \; \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{cp(}\mathit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{eqs\_property} &::= \; \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{termpair\_property} &::= \; \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}
$$

random

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
-

$$
\begin{aligned}
\mathit{strategy} &::= \texttt{?} \mid \texttt{(}\mathit{node\_property}\,\texttt{,}\,\mathit{strategy}\texttt{)} \\
&\quad \mid \mathit{float}\,\texttt{(}\mathit{strategy}\,\texttt{:}\,\mathit{strategy}\texttt{)} \\
\mathit{node\_property} &::= \texttt{*} \mid \texttt{data(}\mathit{termpair\_property}\texttt{)} \mid \texttt{el(}\mathit{pset\_property}\texttt{)} \\
&\quad \mid \texttt{-}\ \mathit{node\_property} \mid \mathit{node\_property}\ \texttt{+}\ \mathit{node\_property} \\
\mathit{pset\_property} &::= \texttt{\#} \mid \texttt{sum(}\mathit{process\_property}\texttt{)} \mid \texttt{min(}\mathit{process\_property}\texttt{)} \\
\mathit{process\_property} &::= \texttt{e(}\mathit{eqs\_property}\texttt{)} \mid \texttt{r(}\mathit{trs\_property}\texttt{)} \mid \texttt{c(}\mathit{trs\_property}\texttt{)} \\
&\quad \mid \mathit{process\_property}\ \texttt{+}\ \mathit{process\_property} \\
\mathit{trs\_property} &::= \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{cp(}\mathit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{eqs\_property} &::= \texttt{sum(}\mathit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\mathit{termpair\_property} &::= \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}
$$

tuple of node properties is compared lexicographically, minimum chosen

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
-

$$
\begin{aligned}
\textit{strategy} &::= \texttt{?} \mid (\textit{node\_property}\,\texttt{,}\,\textit{strategy}) \\
&\quad \mid \textit{float}\,\texttt{(}\textit{strategy}\,\texttt{:}\,\textit{strategy}\texttt{)} \\
\textit{node\_property} &::= \texttt{*} \mid \texttt{data(}\textit{termpair\_property}\texttt{)} \mid \texttt{el(}\textit{pset\_property}\texttt{)} \\
&\quad \mid \texttt{-}\ \textit{node\_property} \mid \textit{node\_property}\ \texttt{+}\ \textit{node\_property} \\
\textit{pset\_property} &::= \texttt{\#} \mid \texttt{sum(}\textit{process\_property}\texttt{)} \mid \texttt{min(}\textit{process\_property}\texttt{)} \\
\textit{process\_property} &::= \texttt{e(}\textit{eqs\_property}\texttt{)} \mid \texttt{r(}\textit{trs\_property}\texttt{)} \mid \texttt{c(}\textit{trs\_property}\texttt{)} \\
&\quad \mid \textit{process\_property}\ \texttt{+}\ \textit{process\_property} \\
\textit{trs\_property} &::= \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{cp(}\textit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\textit{eqs\_property} &::= \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\textit{termpair\_property} &::= \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}
$$

$p(s_1 : s_2)$ takes s_1 with probability p

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process

-

$$
\begin{aligned}
\textit{strategy} &::= \texttt{?} \mid (\textit{node\_property}\texttt{,}\textit{strategy}) \\
&\quad \mid \textit{float}\texttt{(}\textit{strategy}\texttt{:}\textit{strategy}\texttt{)} \\
\textit{node\_property} &::= \texttt{*} \mid \texttt{data(}\textit{termpair\_property}\texttt{)} \mid \texttt{el(}\textit{pset\_property}\texttt{)} \\
&\quad \mid \texttt{-}\ \textit{node\_property} \mid \textit{node\_property}\ \texttt{+}\ \textit{node\_property} \\
\textit{pset\_property} &::= \texttt{\#} \mid \texttt{sum(}\textit{process\_property}\texttt{)} \mid \texttt{min(}\textit{process\_property}\texttt{)} \\
\textit{process\_property} &::= \texttt{e(}\textit{eqs\_property}\texttt{)} \mid \texttt{r(}\textit{trs\_property}\texttt{)} \mid \texttt{c(}\textit{trs\_property}\texttt{)} \\
&\quad \mid \textit{process\_property}\ \texttt{+}\ \textit{process\_property} \\
\textit{trs\_property} &::= \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{cp(}\textit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\textit{eqs\_property} &::= \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\textit{termpair\_property} &::= \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}
$$

Example (size-age ratio)

`0.9((data(sumsize),?):(*,?))`

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
- $$\begin{aligned}
\textit{strategy} &::= \texttt{?} \mid \texttt{(}\textit{node_property}\,\texttt{,}\textit{strategy}\texttt{)} \\
&\mid \textit{float}\,\texttt{(}\textit{strategy}\,\texttt{:}\,\textit{strategy}\texttt{)} \\
\textit{node_property} &::= \texttt{*} \mid \texttt{data(}\textit{termpair_property}\texttt{)} \mid \texttt{el(}\textit{pset_property}\texttt{)} \\
&\mid \texttt{-}\ \textit{node_property} \mid \textit{node_property}\ \texttt{+}\ \textit{node_property} \\
\textit{pset_property} &::= \texttt{\#} \mid \texttt{sum(}\textit{process_property}\texttt{)} \mid \texttt{min(}\textit{process_property}\texttt{)} \\
\textit{process_property} &::= \texttt{e(}\textit{eqs_property}\texttt{)} \mid \texttt{r(}\textit{trs_property}\texttt{)} \mid \texttt{c(}\textit{trs_property}\texttt{)} \\
&\mid \textit{process_property}\ \texttt{+}\ \textit{process_property} \\
\textit{trs_property} &::= \texttt{sum(}\textit{termpair_property}\texttt{)} \mid \texttt{cp(}\textit{eqs_property}\texttt{)} \mid \texttt{\#} \\
\textit{eqs_property} &::= \texttt{sum(}\textit{termpair_property}\texttt{)} \mid \texttt{\#} \\
\textit{termpair_property} &::= \texttt{sizemax} \mid \texttt{sizesum}
\end{aligned}$$

Example (*sum*)

```
(el(min(e(sum(sizesum))+c(sum(sizesum)))),(data(sizesum),(-el(#),?)))
```

Selection Strategies

- first version of mkbTT:
  - select process for which $|E_p(\mathcal{N})| + |R_p(\mathcal{N})|$ is minimal
  - select sometimes old, sometimes small node for process
-

$$
\begin{array}{rcl}
\textit{strategy} & ::= & \texttt{?} \mid \texttt{(}\textit{node\_property}\texttt{,}\textit{strategy}\texttt{)} \\
 & & \mid \textit{float}\texttt{(}\textit{strategy}\texttt{:}\textit{strategy}\texttt{)} \\
\textit{node\_property} & ::= & \texttt{*} \mid \texttt{data(}\textit{termpair\_property}\texttt{)} \mid \texttt{el(}\textit{pset\_property}\texttt{)} \\
 & & \mid \texttt{-}\ \textit{node\_property} \mid \textit{node\_property}\ \texttt{+}\ \textit{node\_property} \\
\textit{pset\_property} & ::= & \texttt{\#} \mid \texttt{sum(}\textit{process\_property}\texttt{)} \mid \texttt{min(}\textit{process\_property}\texttt{)} \\
\textit{process\_property} & ::= & \texttt{e(}\textit{eqs\_property}\texttt{)} \mid \texttt{r(}\textit{trs\_property}\texttt{)} \mid \texttt{c(}\textit{trs\_property}\texttt{)} \\
 & & \mid \textit{process\_property}\ \texttt{+}\ \textit{process\_property} \\
\textit{trs\_property} & ::= & \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{cp(}\textit{eqs\_property}\texttt{)} \mid \texttt{\#} \\
\textit{eqs\_property} & ::= & \texttt{sum(}\textit{termpair\_property}\texttt{)} \mid \texttt{\#} \\
\textit{termpair\_property} & ::= & \texttt{sizemax} \mid \texttt{sizesum}
\end{array}
$$

Example (*max*)

```
(el(min(e(sum(sizemax))+c(sum(sizemax)))),(data(sizemax),(-el(#),?))))
```

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165

(1) time in seconds

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165

(1) time in seconds
(2) number of iterations

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165
CGE_3	∞		215	56	∞		∞		∞	

(1) time in seconds
(2) number of iterations

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165
CGE_3	∞		215	56	∞		∞		∞	
D_8	302	220	∞		∞		∞		∞	

(1) time in seconds
(2) number of iterations

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165
CGE_3	∞		215	56	∞		∞		∞	
D_8	302	220	∞		∞		∞		∞	
ASK93-2	∞		∞		∞		281	314	∞	
SK90-3.04	79	133	1.7	38	17	54	∞		57	126

(1) time in seconds
(2) number of iterations

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165
CGE_3	∞		215	56	∞		∞		∞	
D_8	302	220	∞		∞		∞		∞	
ASK93-2	∞		∞		∞		281	314	∞	
SK90-3.04	79	133	1.7	38	17	54	∞		57	126

(1) time in seconds
(2) number of iterations

Experiments

	sum		max		slothrop		sa		old	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
CGE_2	138	157	9	43	∞		∞		138	165
CGE_3	∞		215	56	∞		∞		∞	
D_8	302	220	∞		∞		∞		∞	
ASK93-2	∞		∞		∞		281	314	∞	
SK90-3.04	79	133	1.7	38	17	54	∞		57	126
⋮	⋮		⋮		⋮		⋮		⋮	
# successes	74		71		57		46		77	
time	17863		18753		27806		33485		16217	

(1) time in seconds
(2) number of iterations

Outline

- Indexing Techniques
- Selection Strategies
- Critical Pair Criteria
- Isomorphisms
- Conclusion

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

deduce $$\frac{\mathcal{N}}{\mathcal{N} \cup \{\langle s : t, \varnothing, \varnothing, R \cap R', \varnothing, \varnothing\rangle\}}$$

if $\langle l : r, R, \ldots\rangle, \langle l' : r', R', \ldots\rangle \in \mathcal{N}$ and $s \leftarrow_{l \to r} u \to_{l' \to r'} t$

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

deduce $\dfrac{\mathcal{N}}{\mathcal{N} \cup \{\langle s : t, \varnothing, \varnothing, R \cap R', \varnothing, \varnothing \rangle\}}$

if $\langle l : r, R, \ldots \rangle, \langle l' : r', R', \ldots \rangle \in \mathcal{N}$
and $s \approx t$ is critical pair

Definition

$\langle l \to r, p, l' \to r' \rangle$ is overlap if

- $p \in \mathrm{Pos}_{\mathcal{F}}(l)$
- mgu σ unifies $l|_p$ and l'
- if $p = \epsilon$ then $l \to r$ and $l' \to r'$ are not variants

Definition

$\langle l \to r, p, l' \to r' \rangle$ is overlap if

- $p \in \mathrm{Pos}_{\mathcal{F}}(l)$
- mgu σ unifies $l|_p$ and l'
- if $p = \epsilon$ then $l \to r$ and $l' \to r'$ are not variants

peak $r\sigma \leftarrow l\sigma \to l\sigma[r'\sigma]_p$ gives rise to critical pair $r\sigma \approx l\sigma[r'\sigma]_p$

Definition

$\langle l \to r, p, l' \to r' \rangle$ is overlap if

- $p \in \mathrm{Pos}_{\mathcal{F}}(l)$
- mgu σ unifies $l|_p$ and l'
- if $p = \epsilon$ then $l \to r$ and $l' \to r'$ are not variants

peak $r\sigma \leftarrow l\sigma \to l\sigma[r'\sigma]_p$ gives rise to critical pair $r\sigma \approx l\sigma[r'\sigma]_p$

Example

$$\mathcal{R}: \quad \begin{aligned} \sqrt{-x+x} &\to 0 && (1)\\ -0+0 &\to 0 && (2)\\ -0 &\to 0 && (3) \end{aligned}$$

$$CP(\mathcal{R}): \quad \begin{aligned} 0 \leftarrow \sqrt{-0+0} &\to \sqrt{0} && \text{from } \langle (1), 1, (2) \rangle\\ 0 \leftarrow \sqrt{-0+0} &\to \sqrt{0+0} && \text{from } \langle (1), 11, (3) \rangle\\ 0 \leftarrow -0+0 &\to \sqrt{0} && \text{from } \langle (2), 1, (3) \rangle \end{aligned}$$

Definition

$\langle l \to r, p, l' \to r' \rangle$ is overlap if

- $p \in \mathcal{P}os_{\mathcal{F}}(l)$
- mgu σ unifies $l|_p$ and l'
- if $p = \epsilon$ then $l \to r$ and $l' \to r'$ are not variants

peak $r\sigma \leftarrow l\sigma \to l\sigma[r'\sigma]_p$ gives rise to critical pair $r\sigma \approx l\sigma[r'\sigma]_p$

Example

$$\mathcal{R}: \quad \begin{array}{rl} \sqrt{-x+x} \to 0 & (1) \\ -0+0 \to 0 & (2) \\ -0 \to 0 & (3) \end{array}$$

all critical pairs required in deduction?

$$CP(\mathcal{R}): \quad \begin{array}{ll} 0 \leftarrow \sqrt{-0+0} \to \sqrt{0} & \text{from } \langle (1), 1, (2) \rangle \\ 0 \leftarrow \sqrt{-0+0} \to \sqrt{0+0} & \text{from } \langle (1), 11, (3) \rangle \\ 0 \leftarrow -0+0 \to \sqrt{0} & \text{from } \langle (2), 1, (3) \rangle \end{array}$$

Critical Pair Criteria in KB

Definition

- CPC is mapping such that $\mathrm{CPC}(\mathcal{E}) = \mathcal{E}'$ where $\mathcal{E}' \subseteq \mathcal{E}$

Critical Pair Criteria in KB

Definition

- CPC is mapping such that $\mathrm{CPC}(\mathcal{E}) = \mathcal{E}'$ where $\mathcal{E}' \subseteq \mathcal{E}$

redundant CPs

Critical Pair Criteria in KB

Definition

- CPC is mapping such that $\mathsf{CPC}(\mathcal{E}) = \mathcal{E}'$ where $\mathcal{E}' \subseteq \mathcal{E}$
- $$\mathcal{E}_0, \mathcal{R}_0 \vdash_{\mathsf{KB}} \mathcal{E}_1, \mathcal{R}_1 \vdash_{\mathsf{KB}} \mathcal{E}_2, \mathcal{R}_2 \vdash_{\mathsf{KB}} \cdots$$

  is fair with respect to CPC if all equations in $\mathsf{CP}(\mathcal{R}_\infty) \setminus \mathsf{CPC}(\bigcup_i \mathcal{E}_i \cup \mathcal{R}_i)$ are deduced

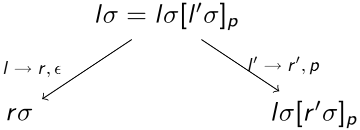
Critical Pair Criteria in KB

Definition

- CPC is mapping such that $\mathrm{CPC}(\mathcal{E}) = \mathcal{E}'$ where $\mathcal{E}' \subseteq \mathcal{E}$
- $$\mathcal{E}_0, \mathcal{R}_0 \vdash_{\mathsf{KB}} \mathcal{E}_1, \mathcal{R}_1 \vdash_{\mathsf{KB}} \mathcal{E}_2, \mathcal{R}_2 \vdash_{\mathsf{KB}} \cdots$$
  is fair with respect to CPC if all equations in $\mathrm{CP}(\mathcal{R}_\infty) \setminus \mathrm{CPC}(\bigcup_i \mathcal{E}_i \cup \mathcal{R}_i)$ are deduced
- CPC is correct if for nonfailing derivation $\mathcal{S}$
  $$\mathcal{S} \text{ is fair with respect to CPC} \Longrightarrow \mathcal{S} \text{ is fair}$$

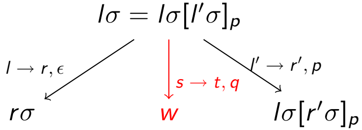
The Primality Criterion

Definition



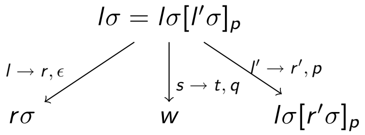
The Primality Criterion

Definition



The Primality Criterion

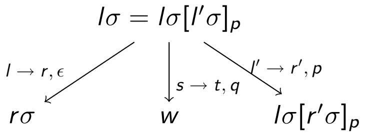
Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

The Primality Criterion

Definition



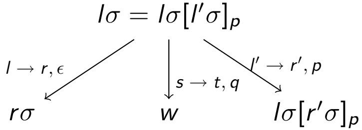
critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

Example

$$\sqrt{-x + x} \to 0 \qquad (1)$$
$$-0 + 0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$

The Primality Criterion

Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

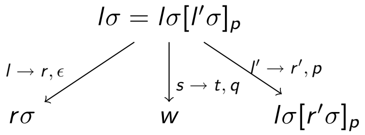
Example

$$\sqrt{-x + x} \to 0 \qquad (1)$$
$$-0 + 0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$

$$\sqrt{-0 + 0}$$

The Primality Criterion

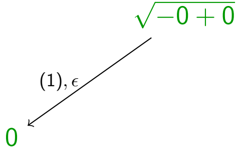
Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

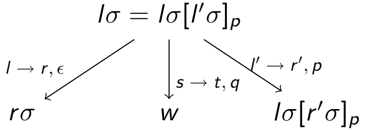
Example

$$\sqrt{-x + x} \to 0 \qquad (1)$$
$$-0 + 0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$



The Primality Criterion

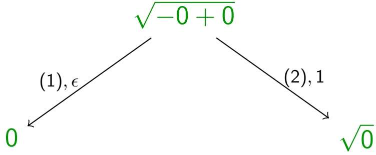
Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

Example

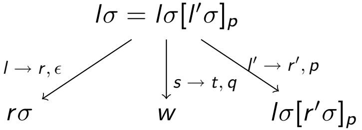
$$\sqrt{-x + x} \to 0 \qquad (1)$$
$$-0 + 0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$



Critical pair $0 \approx \sqrt{0}$

The Primality Criterion

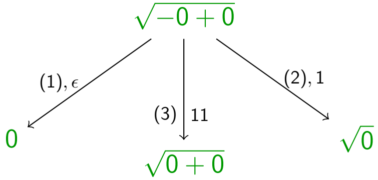
Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

Example

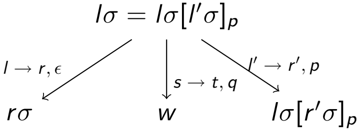
$$\sqrt{-x + x} \to 0 \qquad (1)$$
$$-0 + 0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$



Critical pair $0 \approx \sqrt{0}$

The Primality Criterion

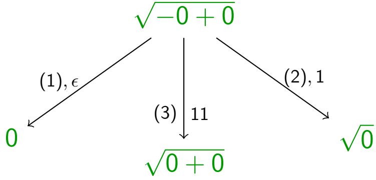
Definition



critical pair $r\sigma \approx l\sigma[v\sigma]_p$ is composite if $p < q$

Example

$$\sqrt{-x+x} \to 0 \qquad (1)$$
$$-0+0 \to 0 \qquad (2)$$
$$-0 \to 0 \qquad (3)$$



Critical pair $0 \approx \sqrt{0}$ is composite

Critical Pair Criteria in mkbTT

Definition

- if $s \approx t$ is CP then $\mathsf{CPC_m}$ is mapping such that $\mathrm{CPC}_m(s \approx t, P) = P'$ where $P' \subseteq P$

Critical Pair Criteria in mkbTT

Definition

- if $s \approx t$ is CP then CPC$_\mathrm{m}$ is mapping such that $\mathrm{CPC}_m(s \approx t, P) = P'$ where $P' \subseteq P$ processes for which CP is redundant

Critical Pair Criteria in mkbTT

Definition

- if $s \approx t$ is CP then $\mathrm{CPC_m}$ is mapping such that $\mathrm{CPC}_m(s \approx t, P) = P'$ where $P' \subseteq P$
- $$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \ldots \vdash \mathcal{N}_k$$

  is fair with respect to CPC_m if projection is fair with respect to CPC for some process $p \in \mathcal{N}_k$

Critical Pair Criteria in mkbTT

Definition

- if $s \approx t$ is CP then $\mathrm{CPC_m}$ is mapping such that $\mathrm{CPC}_m(s \approx t, P) = P'$ where $P' \subseteq P$
- $$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \ldots \vdash \mathcal{N}_k$$
  is fair with respect to CPC_m if projection is fair with respect to CPC for some process $p \in \mathcal{N}_k$
- obtain $\mathrm{CPC_m}$ from CPC by setting $\mathrm{CPC}_m(s \approx t, P) = Q$ such that $s \approx t \in \mathrm{CPC}(E_p(\mathcal{N}))$ for all $p \in Q$

Critical Pair Criteria in mkbTT

Definition

- if $s \approx t$ is CP then $\mathrm{CPC_m}$ is mapping such that $\mathrm{CPC}_m(s \approx t, P) = P'$ where $P' \subseteq P$
-
$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \ldots \vdash \mathcal{N}_k$$
is fair with respect to CPC_m if projection is fair with respect to CPC for some process $p \in \mathcal{N}_k$
- obtain $\mathrm{CPC_m}$ from CPC by setting
$\mathrm{CPC}_m(s \approx t, P) = Q$ such that $s \approx t \in \mathrm{CPC}(E_p(\mathcal{N}))$ for all $p \in Q$

Lemma

if CPC is correct then also CPC_m is correct

Implementation

- primality criterion PCP — Kapur et al '88
- blocking criterion BCP — Bachmair/Dershowitz '88
- connectedness criterion CCP — Küchlin '85

Implementation

exploit sharing

- primality criterion PCP — Kapur et al '88
- blocking criterion BCP — Bachmair/Dershowitz '88
- connectedness criterion CCP — Küchlin '85

Implementation

- primality criterion PCP — Kapur et al '88
- blocking criterion BCP — Bachmair/Dershowitz '88
- connectedness criterion CCP — Küchlin '85

Experiments

none		PCP			BCP			CCP			all		
(1)	(2)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
70	189	71	188	416	71	188	391	70	189	155	71	188	468

(1) number of successful completions
(2) average time
(3) number of redundant critical pairs for successful process

Outline

- Indexing Techniques
- Selection Strategies
- Critical Pair Criteria
- Isomorphisms
- Conclusion

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
           N_o := rewrite(N_c, {n}) ∪ N_o
           N_o := deduce(n, N_c) ∪ N_o
           N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

```
procedure mkbTT(N_o, N_c)
if success then
    return p
else if N_o = ∅ then
    fail
else
    n := choose(N_o)
    N_o := rewrite({n}, N_c) ∪ (N_o \ {n})
    if n ≠ ⟨..., ∅, ∅, ∅, ∅, ∅⟩ then
        n := orient(n)
        if n ≠ ⟨..., ∅, ∅, ..., ..., ...⟩ then
            N_o := rewrite(N_c, {n}) ∪ N_o
            N_o := deduce(n, N_c) ∪ N_o
            N_c := N_c ∪ {n}
mkbTT(N_o, N_c)
```

Example (Renaming)

- mkbTT run on CGE_2

$\mathcal{N}_0$

$$E_\epsilon = \begin{cases} (x * y) * z \approx x * (y * z) & i(x) * x \approx 1 \\ x * 1 \approx 1 & g(x) * f(y) \approx f(y) * g(x) \\ f(x * y) \approx f(x) * f(y) & g(x * y) \approx g(x) * g(y) \end{cases} \qquad R_\epsilon = C_\epsilon = \varnothing$$

Example (Renaming)

- mkbTT run on CGE_2

$\mathcal{N}_0 \vdash \mathcal{N}_1$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2$$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i$$

$$E_p = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \\ g(x) * f(y) \approx f(y) * g(x) \end{cases} \qquad R_p = C_p = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \end{cases}$$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash$$

$$E_p = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \\ g(x) * f(y) \approx f(y) * g(x) \end{cases} \qquad R_p = C_p = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \end{cases}$$

$$\text{orient } \frac{\langle f(x) * g(y) : g(y) * f(x), \varnothing, \varnothing, \{p\}, \varnothing, \varnothing \rangle}{\langle f(x) * g(y) : g(y) * f(x), \{p0\}, \{p1\}, \varnothing, \varnothing, \varnothing \rangle}$$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \quad R_{p0} = C_{p0} = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \\ g(x) * f(y) \to f(y) * g(x) \end{cases}$$

$$E_{p1} = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \quad R_{p1} = C_{p1} = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \\ f(y) * g(x) \to g(x) * f(y) \end{cases}$$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \qquad R_{p0} = C_{p0} = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \\ g(x) * f(y) \to f(y) * g(x) \end{cases}$$

$$E_{p1} = \begin{cases} (x * y) * z \approx x * (y * z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \qquad R_{p1} = C_{p1} = \begin{cases} 1 * x \to x \\ i(x) * x \to 1 \\ f(x * y) \to f(x) * f(y) \\ g(x * y) \to g(x) * g(y) \\ f(y) * g(x) \to g(x) * f(y) \end{cases}$$

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta\colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathcal{T}(\mathcal{F},\mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

- $\mathcal{R}' = \{\theta(l) \to \theta(r) \mid l \to r \in \mathcal{R}\}$,

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta \colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

- $\mathcal{R}' = \{\theta(l) \to \theta(r) \mid l \to r \in \mathcal{R}\}$,
- $|\mathcal{R}| = |\mathcal{R}'|$ and

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta\colon \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

- $\mathcal{R}' = \{\theta(l) \to \theta(r) \mid l \to r \in \mathcal{R}\}$,
- $|\mathcal{R}| = |\mathcal{R}'|$ and
- $\forall s, t \qquad s \to_{\mathcal{R}} t \quad$ if and only if $\quad \theta(s) \to_{\mathcal{R}'} \theta(t)$

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathcal{T}(\mathcal{F},\mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

- $\mathcal{R}' = \{\theta(l) \to \theta(r) \mid l \to r \in \mathcal{R}\}$,
- $|\mathcal{R}| = |\mathcal{R}'|$ and
- $\forall s, t \qquad s \to_\mathcal{R} t \quad$ if and only if $\quad \theta(s) \to_{\mathcal{R}'} \theta(t)$

Definition

processes p, q are isomorphic in $\mathcal{N}$ if for some θ

$$R_p(\mathcal{N}) \cong_\theta R_q(\mathcal{N}) \qquad C_p(\mathcal{N}) \cong_\theta C_q(\mathcal{N}) \qquad E_p(\mathcal{N}) \cong_\theta E_q(\mathcal{N})$$

Definition

for rewrite systems $\mathcal{R}, \mathcal{R}'$, mapping $\theta\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathcal{T}(\mathcal{F},\mathcal{V})$ induces isomorphism

$$\mathcal{R} \cong_\theta \mathcal{R}'$$

if

- $\mathcal{R}' = \{\theta(l) \to \theta(r) \mid l \to r \in \mathcal{R}\}$,
- $|\mathcal{R}| = |\mathcal{R}'|$ and
- $\forall s, t \qquad s \to_{\mathcal{R}} t \quad$ if and only if $\quad \theta(s) \to_{\mathcal{R}'} \theta(t)$

Definition

processes p, q are isomorphic in $\mathcal{N}$ if for some θ

$$R_p(\mathcal{N}) \cong_\theta R_q(\mathcal{N}) \qquad C_p(\mathcal{N}) \cong_\theta C_q(\mathcal{N}) \qquad E_p(\mathcal{N}) \cong_\theta E_q(\mathcal{N})$$

Lemma

if p, q are isomorphic in $\mathcal{N}$ then
$\exists \mathcal{N}' \;\; \mathcal{N} \vdash^* \mathcal{N}'$ with $E_p(\mathcal{N}') = \varnothing \quad \Longleftrightarrow \quad \exists \mathcal{N}'' \;\; \mathcal{N} \vdash^* \mathcal{N}''$ and $E_q(\mathcal{N}'') = \varnothing$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \cdots \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \begin{cases} (x*y)*z \approx x*(y*z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \quad R_{p0} = C_{p0} = \begin{cases} 1*x \to x \\ i(x)*x \to 1 \\ f(x*y) \to f(x)*f(y) \\ g(x*y) \to g(x)*g(y) \\ g(x)*f(y) \to f(y)*g(x) \end{cases}$$

$$E_{p1} = \begin{cases} (x*y)*z \approx x*(y*z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \quad R_{p1} = C_{p1} = \begin{cases} 1*x \to x \\ i(x)*x \to 1 \\ f(x*y) \to f(x)*f(y) \\ g(x*y) \to g(x)*g(y) \\ f(y)*g(x) \to g(x)*f(y) \end{cases}$$

- $p0, p1$ are isomorphic using $\theta(t) = \begin{cases} f(t') & \text{if } t = g(t') \\ g(t') & \text{if } t = f(t') \\ t & \textit{otherwise} \end{cases}$

Example (Renaming)

- mkbTT run on CGE_2

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \begin{cases} (x*y)*z \approx x*(y*z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \qquad R_{p0} = C_{p0} = \begin{cases} 1*x \to x \\ i(x)*x \to 1 \\ f(x*y) \to f(x)*f(y) \\ g(x*y) \to g(x)*g(y) \\ g(x)*f(y) \to f(y)*g(x) \end{cases}$$

$$E_{p1} = \begin{cases} (x*y)*z \approx x*(y*z) \\ f(1) \approx 1 \\ g(1) \approx 1 \end{cases} \qquad R_{p1} = C_{p1} = \begin{cases} 1*x \to x \\ i(x)*x \to 1 \\ f(x*y) \to f(x)*f(y) \\ g(x*y) \to g(x)*g(y) \\ f(y)*g(x) \to g(x)*f(y) \end{cases}$$

- $p0, p1$ are isomorphic using $\theta(t) = \begin{cases} f(t') & \text{if } t = g(t') \\ g(t') & \text{if } t = f(t') \\ t & \textit{otherwise} \end{cases}$
- join $p0, p1$

Example (Argument Permutations)

$\mathcal{N}_0$

$$E_\epsilon = \begin{cases} f(f(x)) \approx x \\ f(x+y) \approx f(x) + f(y) \\ x + (y+z) \approx (x+y)+z \end{cases} \qquad R_\epsilon = C_\epsilon = \varnothing$$

Example (Argument Permutations)

$\mathcal{N}_0 \vdash \mathcal{N}_1$

Example (Argument Permutations)

$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2$

Example (Argument Permutations)

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i$$

$$E_p = \{x + (y + z) \approx (x + y) + z \qquad R_p = C_p = \begin{cases} f(f(x)) \to x \\ f(x + y) \to f(x) + f(y) \end{cases}$$

Example (Argument Permutations)

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash$$

$E_p = \{x + (y + z) \approx (x + y) + z$ $\quad R_p = C_p = \begin{cases} f(f(x)) \to x \\ f(x+y) \to f(x) + f(y) \end{cases}$

$$\text{orient}\ \frac{\langle x + (y + z) : (x + y) + z, \varnothing, \varnothing, \{p\}, \varnothing, \varnothing \rangle}{\langle x + (y + z) : (x + y) + z, \{p0\}, \{p1\}, \varnothing, \varnothing, \varnothing \rangle}$$

Example (Argument Permutations)

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \quad \cdots \quad \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \varnothing \quad R_{p0} = C_{p0} = \begin{cases} f(f(x)) \to x \\ f(x+y) \to f(x) + f(y) \\ x + (y + z) \to (x + y) + z \end{cases}$$

$$E_{p1} = \varnothing \quad R_{p1} = C_{p1} = \begin{cases} f(f(x)) \to x \\ f(x+y) \to f(x) + f(y) \\ (x + y) + z \to x + (y + z) \end{cases}$$

Example (Argument Permutations)

$$\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \cdots \vdash \mathcal{N}_i \vdash \mathcal{N}_{i+1}$$

$$E_{p0} = \varnothing \quad R_{p0} = C_{p0} = \begin{cases} f(f(x)) \to x \\ f(x+y) \to f(x)+f(y) \\ x+(y+z) \to (x+y)+z \end{cases}$$

$$E_{p1} = \varnothing \quad R_{p1} = C_{p1} = \begin{cases} f(f(x)) \to x \\ f(x+y) \to f(x)+f(y) \\ (x+y)+z \to x+(y+z) \end{cases}$$

- $p0, p1$ are isomorphic using $\theta(t) = \begin{cases} s+u & \text{if } t = u+s \\ t & \textit{otherwise} \end{cases}$

Implementation

- isomorphism checks for renamings or argument permutations

Implementation

- isomorphism checks for renamings or argument permutations

Experiments

- renamings allow to complete
    - CGE_2 in 4.7 (instead of 8.4), CGE_3 in 33 (instead of 184)

Implementation

- isomorphism checks for renamings or argument permutations

Experiments

- renamings allow to complete
  - CGE_2 in 4.7 (instead of 8.4), CGE_3 in 33 (instead of 184)
  - CGE_4 in 622 seconds

Implementation

- isomorphism checks for renamings or argument permutations

Experiments

- renamings allow to complete
  - CGE_2 in 4.7 (instead of 8.4), CGE_3 in 33 (instead of 184)
  - CGE_4in 622 seconds

none			renamings			permutations		
(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
18774	10.9	25.2	18634	8.9	21.1	18841	11.8	25.0

(1) total time
(2) average time for successful completion
(3) average number of processes

Conclusion

Usage

various options can be controlled via web interface

Conclusion

Usage

various options can be controlled via web interface

Conclusion

- indexing techniques pay off
- selection strategies have great impact – optimal one?
- critical pair criteria allow for small improvements
- renaming isomorphisms are very useful for special systems