Finding and Certifying Loops

Harald Zankl Christian Sternagel Dieter Hofbauer Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria
Berufsakademie Nordhessen, Bad Wildungen, Germany

SOFSEM 2010 27 January 2010

Overview

- String Rewriting
- Finding Loops
  - SAT
  - Closures
- Certifying Loops
- Experiments
- Conclusion

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

abb

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa}$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

Definition

An SRS $\mathcal{S}$ is terminating if there is no infinite sequence

$$t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} t_3 \to_{\mathcal{S}} \cdots$$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

Definition

An SRS $\mathcal{S}$ is terminating if there is no infinite sequence

$$t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} t_3 \to_{\mathcal{S}} \cdots$$

An SRS $\mathcal{S}$ is called looping if $t_1 \to_{\mathcal{S}} t_2 \to \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} u t_1 v$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

Definition

An SRS $\mathcal{S}$ is terminating if there is no infinite sequence

$$t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} t_3 \to_{\mathcal{S}} \cdots$$

An SRS $\mathcal{S}$ is called looping if $t_1 \to_{\mathcal{S}} t_2 \to \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} u t_1 v$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

Definition

An SRS $\mathcal{S}$ is terminating if there is no infinite sequence

$$t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} t_3 \to_{\mathcal{S}} \cdots$$

An SRS $\mathcal{S}$ is called looping if $t_1 \to_{\mathcal{S}} t_2 \to \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} u t_1 v$

String Rewriting

Example (String Rewriting System (SRS))

$\mathcal{S} = \{\, \mathsf{ab} \to \mathsf{bbaa} \}$

$$\mathsf{abb} \to_{\mathcal{S}} \mathsf{bbaab} \to_{\mathcal{S}} \mathsf{bbabbaa} \to_{\mathcal{S}} \mathsf{bbbbaabaa} \to_{\mathcal{S}} \cdots$$

Definition

An SRS $\mathcal{S}$ is terminating if there is no infinite sequence

$$t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} t_3 \to_{\mathcal{S}} \cdots$$

An SRS $\mathcal{S}$ is called looping if $t_1 \to_{\mathcal{S}} t_2 \to \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} u t_1 v$

Lemma

If an SRS $\mathcal{S}$ is looping then it is not terminating

Finding Loops with SAT

abb	$\rightarrow_{\mathcal{S}}$	M^{a}_{11}	M^{b}_{12}	M^{b}_{13}				
bbaab	$\rightarrow_{\mathcal{S}}$	M^{b}_{21}	M^{b}_{22}	M^{a}_{23}	M^{a}_{24}	M^{b}_{25}		
bbabbaa	$\rightarrow_{\mathcal{S}}$	M^{b}_{31}	M^{b}_{32}	M^{a}_{33}	M^{b}_{34}	M^{b}_{35}	M^{a}_{36}	M^{a}_{37}
$\ldots$	$\rightarrow_{\mathcal{S}}$	.	.	.	.	.	.	.

SAT Encoding

- matrix M with m rows and n columns
- M^{a}_{ij} is true iff $M_{(ij)} = \mathsf{a}$

Finding Loops with SAT

abb	$\to_{\mathcal{S}}$	M^{a}_{11}	M^{b}_{12}	M^{b}_{13}				
bbaab	$\to_{\mathcal{S}}$	M^{b}_{21}	M^{b}_{22}	M^{a}_{23}	M^{a}_{24}	M^{b}_{25}		
bbabbaa	$\to_{\mathcal{S}}$	M^{b}_{31}	M^{b}_{32}	M^{a}_{33}	M^{b}_{34}	M^{b}_{35}	M^{a}_{36}	M^{a}_{37}
$\dots$	$\to_{\mathcal{S}}$	.	.	.	.	.	.	.

Constraints for

- for every i, j exactly one M^a_{ij} is true
  $\bigwedge_{i,j}(\bigvee_a M^a_{ij}) \wedge (\bigwedge_a M^a_{ij} \to \bigwedge_{b\neq a} \neg M^b_{ij})$
- constraints for applying rule correctly
- constraint for loop (empty context due to DP transformation)
  $\bigvee_{i>1}(\bigwedge_{j,a}(M^a_{1j} \leftrightarrow M^a_{ij}))$

Finding Loops with SAT

abb	$\to_{\mathcal{S}}$	M^{a}_{11}	M^{b}_{12}	M^{b}_{13}				
bbaab	$\to_{\mathcal{S}}$	M^{b}_{21}	M^{b}_{22}	M^{a}_{23}	M^{a}_{24}	M^{b}_{25}		
bbabbaa	$\to_{\mathcal{S}}$	M^{b}_{31}	M^{b}_{32}	M^{a}_{33}	M^{b}_{34}	M^{b}_{35}	M^{a}_{36}	M^{a}_{37}
$\dots$	$\to_{\mathcal{S}}$	.	.	.	.	.	.	.

Constraints for

- for every i, j exactly one M^{a}_{ij} is true
  $\bigwedge_{i,j}(\bigvee_{a} M^{a}_{ij}) \wedge (\bigwedge_{a} M^{a}_{ij} \to \bigwedge_{b\neq a} \neg M^{b}_{ij})$
- constraints for applying rule correctly
- constraint for loop (empty context due to DP transformation)
  $\bigvee_{i>1}(\bigwedge_{j,a}(M^{a}_{1j} \leftrightarrow M^{a}_{ij}))$

Finding Loops with SAT

abb	$\to_{\mathcal{S}}$	M^{a}_{11}	M^{b}_{12}	M^{b}_{13}				
bbaab	$\to_{\mathcal{S}}$	M^{b}_{21}	M^{b}_{22}	M^{a}_{23}	M^{a}_{24}	M^{b}_{25}		
bbabbaa	$\to_{\mathcal{S}}$	M^{b}_{31}	M^{b}_{32}	M^{a}_{33}	M^{b}_{34}	M^{b}_{35}	M^{a}_{36}	M^{a}_{37}
$\dots$	$\to_{\mathcal{S}}$	.	.	.	.	.	.	.

Constraints for

- for every i, j exactly one M^{a}_{ij} is true
  $\bigwedge_{i,j}(\bigvee_{a} M^{a}_{ij}) \wedge (\bigwedge_{a} M^{a}_{ij} \to \bigwedge_{b\neq a} \neg M^{b}_{ij})$
- constraints for applying rule correctly
- constraint for loop (empty context due to DP transformation)
  $\bigvee_{i>1}(\bigwedge_{j,a}(M^{a}_{1j} \leftrightarrow M^{a}_{ij}))$

Finding Loops with SAT

abb	$\rightarrow_{\mathcal{S}}$	M^{a}_{11}	M^{b}_{12}	M^{b}_{13}				
bbaab	$\rightarrow_{\mathcal{S}}$	M^{b}_{21}	M^{b}_{22}	M^{a}_{23}	M^{a}_{24}	M^{b}_{25}		
bbabbaa	$\rightarrow_{\mathcal{S}}$	M^{b}_{31}	M^{b}_{32}	M^{a}_{33}	M^{b}_{34}	M^{b}_{35}	M^{a}_{36}	M^{a}_{37}
$\ldots$	$\rightarrow_{\mathcal{S}}$	.	.	.	.	.	.	.

Remark

implemented in T$_{\mathsf{T}}$T$_2$ (Tyrolean Termination Tool 2)

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathrm{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathsf{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathsf{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathsf{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathsf{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathsf{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathsf{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathsf{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, sr) \in \mathsf{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathrm{RFC}(\mathcal{S})$

Example ($\mathcal{S} = \{ab \to bbaa\}$)

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, sr) \in \mathrm{RFC}(\mathcal{S})$

Example ($\mathcal{S} = \{ab \to bbaa\}$)

- $(ab, bbaa) \in \mathrm{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, sr) \in \mathrm{RFC}(\mathcal{S})$

Example ($\mathcal{S} = \{ab \to bbaa\}$)

- $(ab, bbaa) \in \mathrm{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, sr) \in \mathrm{RFC}(\mathcal{S})$

Example ($\mathcal{S} = \{ab \to bbaa\}$)

- $(ab, bbaa) \in \mathrm{RFC}(\mathcal{S})$
- $(abb, bbaab, bbabbaa) \in \mathrm{RFC}(\mathcal{S})$

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathsf{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathsf{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathsf{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathsf{RFC}(\mathcal{S})$

Lemma (Geser & Zantema, 1999)

$\mathcal{S}$ looping if and only if RFC($\mathcal{S}$) looping

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathrm{RFC}(\mathcal{S})$

Lemma (Geser & Zantema, 1999)

$\mathcal{S}$ looping if and only if RFC($\mathcal{S}$) looping

Remarks

- RFC and LFC implemented in KFL (Knocked-For-Loops)
- most loops are thin (restrict ② steps)

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, sr) \in \mathrm{RFC}(\mathcal{S})$

Lemma (Geser & Zantema, 1999)

$\mathcal{S}$ looping if and only if RFC($\mathcal{S}$) looping

Remarks

- RFC and LFC implemented in KFL (Knocked-For-Loops)
- most loops are thin (restrict ② steps)

Closures

Right Forward Closures

- $\mathcal{S} \subseteq \mathrm{RFC}(\mathcal{S})$
- $(t_1, \ldots, t_n) \in \mathrm{RFC}(\mathcal{S})$ and
  1. $t_n \to_{\mathcal{S}} t_{n+1}$ then $(t_1, \ldots, t_n, t_{n+1}) \in \mathrm{RFC}(\mathcal{S})$
  2. $t_n = s l_1$ and $l_1 l_2 \to r \in \mathcal{S}$ then $(t_1 l_2, \ldots, t_n l_2, s r) \in \mathrm{RFC}(\mathcal{S})$

Lemma (Geser & Zantema, 1999)

$\mathcal{S}$ looping if and only if RFC($\mathcal{S}$) looping

Remarks

- RFC and LFC implemented in KFL (Knocked-For-Loops)
- most loops are thin (restrict ② steps)

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Main Lemma

If $\mathcal{S}$ looping then $\mathcal{S}$ not terminating

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Main Lemma

If $\mathcal{S}$ looping then $\mathcal{S}$ not terminating

Formalization

- given: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- duty: prove $\mathcal{S}$ not terminating
- idea: construct sequence
  $t_1 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} ut_nv \to_{\mathcal{S}} uut_1vv \to_{\mathcal{S}} \cdots$
- rewriting is closed under left and right contexts

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Main Lemma

If $\mathcal{S}$ looping then $\mathcal{S}$ not terminating

Formalization

- given: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- duty: prove $\mathcal{S}$ not terminating
- idea: construct sequence
  $t_1 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} ut_nv \to_{\mathcal{S}} uut_1vv \to_{\mathcal{S}} \cdots$
- rewriting is closed under left and right contexts

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Main Lemma

If $\mathcal{S}$ looping then $\mathcal{S}$ not terminating

Formalization

- given: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- duty: prove $\mathcal{S}$ not terminating
- idea: construct sequence
  $t_1 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} ut_nv \to_{\mathcal{S}} uut_1vv \to_{\mathcal{S}} \cdots$
- rewriting is closed under left and right contexts

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $t_i \to_{\mathcal{S}} t_{i+1}$

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $t_i \to_{\mathcal{S}} t_{i+1}$

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $u'lv' = t_i \to_{\mathcal{S}} t_{i+1}$

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $u'lv' = t_i \to_{\mathcal{S}} t_{i+1} = u'rv'$

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $u'lv' = t_i \to_{\mathcal{S}} t_{i+1} = u'rv' \quad l \to r \in \mathcal{S}$

Certifying Loops

Basics

- Isabelle is a higher-order theorem prover
- IsaFoR is an Isabelle formalization of rewriting
- CeTA is a certifier for termination analysis

Certifier

- given: $t_1, t_2, \ldots, t_n, u, v$ and SRS $\mathcal{S}$
- duty: $t_1 \to_{\mathcal{S}} t_2 \to_{\mathcal{S}} \cdots \to_{\mathcal{S}} t_n \to_{\mathcal{S}} ut_1v$
- idea: $u'lv' = t_i \to_{\mathcal{S}} t_{i+1} = u'rv' \quad l \to r \in \mathcal{S}$

Experiments

732 SRSs from TPDB

tool	KFL /CeTA	Matchbox	nonloop	NTI	$T_T T_2$ /CeTA
no	158 / 158	149	93	67	97 / 97
time	34853/ 6	35441	37855	39508	23039/ 3
time (avg.)	2.45 /0.03	3.10	5.73	5.78	11.61 /0.02

Experiments

732 SRSs from TPDB

tool	KFL /CeTA	Matchbox	nonloop	NTI	TTT2 /CeTA
no	158 / 158	149	93	67	97 / 97
time	34853/ 6	35441	37855	39508	23039/ 3
time (avg.)	2.45 /0.03	3.10	5.73	5.78	11.61 /0.02

Remark

- 524 SRSs known to be terminating
- just 41 unknown
- longest loop: `Gebhard/10`, length 80, width 21

Experiments

732 SRSs from TPDB

tool	KFL /CeTA	Matchbox	nonloop	NTI	TTT2 /CeTA
no	158 / 158	149	93	67	97 / 97
time	34853/ 6	35441	37855	39508	23039/ 3
time (avg.)	2.45 /0.03	3.10	5.73	5.78	11.61 /0.02

Remark

- 524 SRSs known to be terminating
- just 41 unknown
- longest loop: `Gebhard/10`, length 80, width 21

Experiments

732 SRSs from TPDB

tool	KFL /CeTA	Matchbox	nonloop	NTI	TTT2 /CeTA
no	158 / 158	149	93	67	97 / 97
time	34853/ 6	35441	37855	39508	23039/ 3
time (avg.)	2.45 /0.03	3.10	5.73	5.78	11.61 /0.02

Remark

- 524 SRSs known to be terminating
- just 41 unknown
- longest loop: `Gebhard/10`, length 80, width 21

Conclusion

Contributions

- 2 methods for finding loops (SAT, closures)
- formalization of loops (IsaFoR)
- certification of loops (CeTA)

Conclusion

Contributions

- 2 methods for finding loops (SAT, closures)
- formalization of loops (IsaFoR)
- certification of loops (CeTA)

Conclusion

Contributions

- 2 methods for finding loops (SAT, closures)
- formalization of loops (IsaFoR)
- certification of loops (CeTA)

Related Work

- international competition of termination tools

  `http://termcomp.uibk.ac.at`

- termination provers + certifiers compete on large testbed

Conclusion

Contributions

- 2 methods for finding loops (SAT, closures)
- formalization of loops (IsaFoR)
- certification of loops (CeTA)

Related Work

- international competition of termination tools

  `http://termcomp.uibk.ac.at`

- termination provers + certifiers compete on large testbed

Future Work

- certification of non-looping non-termination