Towards Complexity Analysis of Programs by Rewriting

Georg Moser

Institute of Computer Science
University of Innsbruck

AJSW 2010, August 1 – 7, 2010

Example

```
public class Flatten {
  public static IntList flatten(TreeList list) {
    TreeList cur = list; IntList result = null;
    while (cur != null) {
      Tree tree = cur.value;
      if (tree != null) {
        IntList oldIntList = result;
        result = new IntList();
        result.value = tree.value;
        result.next = oldIntList;
        TreeList oldCur = cur;
        cur = new TreeList();
        cur.value = tree.left;
        cur.next = oldCur;
        oldCur.value = tree.right;
      } else cur = cur.next;
  } return result;
}
```

Howto Analyse the Complexity of `flatten` Automatically?

C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl.
Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. of 21th RTA*, pages 259-276, LIPIcs, 2010.

Howto Analyse the Complexity of `flatten` Automatically?

C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl.
Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. of 21th RTA*, pages 259-276, LIPIcs, 2010.

Result

- automated technique to prove termination of JBC programs
- transformation of JBC programs (via termination graphs) into TRSs

Howto Analyse the Complexity of `flatten` Automatically?

C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl.
Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. of 21th RTA*, pages 259-276, LIPIcs, 2010.

Result

- automated technique to prove termination of JBC programs
- transformation of JBC programs (via termination graphs) into TRSs

Example

consider the TRS $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

Howto Analyse the Complexity of `flatten` Automatically?

C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl.
Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. of 21th RTA*, pages 259-276, LIPIcs, 2010.

Result

- automated technique to prove termination of JBC programs
- transformation of JBC programs (via termination graphs) into ITRSs

Example

consider the TRS $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

Howto Analyse the Complexity of `flatten` Automatically?

C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl.
Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. of 21th RTA*, pages 259-276, LIPIcs, 2010.

Result has been presented at RTA

- automated technique to prove termination of JBC programs
- transformation of JBC programs (via termination graphs) into ITRSs

Example

consider the TRS $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

Example (cont'd)

re-consider $\mathcal{R}_{\mathsf{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

Example (cont'd)

re-consider $\mathcal{R}_{\mathsf{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call $\mathsf{T}_{\mathcal{C}}\mathsf{T}$ on $\mathcal{R}_{\mathsf{flatten}}$

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, O(n^1))$

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, \mathsf{O}(n^1))$

Assessment

- linear upper-bound on complexity of `flatten` automatically verified

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, \mathsf{O}(n^1))$

Assessment

- linear upper-bound on complexity of `flatten` automatically verified
- but is this certificate really correct?

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x,\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, O(n^1))$

Assessment

easy negative answer

- linear upper-bound on co[illegible] automatically verified
- but is this certificate really correct?

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) \to \mathsf{f}(\mathsf{g}(0,x),x,y)$$
$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) \to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))$$
$$\mathsf{f}(x,\mathsf{g}(0,y),z) \to \mathsf{f}(x,y,z)$$
$$\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) \to \mathsf{f}(x',\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, O(n^1))$

Assessment

easy negative answer

- linear upper-bound on co[illegible] automatically verified
- but is this certificate really correct?

Example (cont'd)

re-consider $\mathcal{R}_{\text{flatten}}$

$$\begin{aligned}
\mathsf{f}(\mathsf{g}(0,x),\mathsf{g}(0,x),y) &\to \mathsf{f}(\mathsf{g}(0,x),x,y)\\
\mathsf{f}(\mathsf{g}(\mathsf{h}(x,y,z),u),\mathsf{g}(\mathsf{h}(x,y,z),u),v) &\to \mathsf{f}(\mathsf{g}(z,u),\mathsf{g}(y,\mathsf{g}(z,u)),\mathsf{i}(x,v))\\
\mathsf{f}(x,\mathsf{g}(0,y),z) &\to \mathsf{f}(x,y,z)\\
\mathsf{f}(x,\mathsf{g}(\mathsf{h}(y,z,u),v),w) &\to \mathsf{f}(x',\mathsf{g}(z,\mathsf{g}(u,v),\mathsf{i}(y,w)))
\end{aligned}$$

- call TCT on $\mathcal{R}_{\text{flatten}}$
- TCT answers YES$(?, O(n^1))$

Assessment

so, let's see ...

- linear upper-bound on com[illegible]ten automatically verified
- but is this certificate really correct?

Challenge ①

is the runtime of `flatten` faithfully represented by the derivation height wrt its TRS representation?

Challenge ①

is the runtime of `flatten` faithfully represented by the derivation height wrt its TRS representation?

Challenge ②

is the runtime complexity of `flatten` faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ①

is the runtime of `flatten` faithfully represented by the derivation height wrt its TRS representation?

Challenge ②

is the runtime complexity of `flatten` faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ③

is the runtime complexity of `flatten` faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Challenge ①

is the runtime of `flatten` faithfully represented by the derivation height wrt its TRS representation?

Challenge ②

is the runtime complexity of `flatten` faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ③

is the runtime complexity of `flatten` faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Challenge ④

is the analysis extendable to TRSs with fresh variables?

Challenge ①

is the runtime of any JBC program faithfully represented by the derivation height wrt its TRS representation?

Challenge ②

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ③

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Challenge ④

is the analysis extendable to ITRSs with fresh variables?

Challenge ①

is the runtime of any JBC program faithfully represented by the derivation height wrt its TRS representation?

Challenge ②

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ③

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Challenge ④

and what about automation?

is the analysis extendable to ITRSs with fresh variables?

Overview

- Preliminaries
- Termination Analysis of Java Programs
- Challenges
  - Challenge ①
  - Challenge ②
  - Challenge ③
  - Challenge ④
- Main Result

Complexity of Term Rewrite Systems

Definition

the derivation height of a term t wrt to T and $\to$

$$\mathrm{dh}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathrm{dh}(n, T, \to) = \max\{\mathrm{dh}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Complexity of Term Rewrite Systems

Definition

the derivation height of a term t wrt to T and $\to$

$$\mathrm{dh}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathrm{dh}(n, T, \to) = \max\{\mathrm{dh}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition

the derivational complexity with respect to a TRS $\mathcal{R}$

$$\mathrm{dc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Complexity of Term Rewrite Systems

Definition

the derivation height of a term t wrt to T and $\to$

$$\mathrm{dh}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathrm{dh}(n, T, \to) = \max\{\mathrm{dh}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition

the derivational complexity with respect to a TRS $\mathcal{R}$

$$\mathrm{dc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Definition

the runtime complexity with respect to a TRS $\mathcal{R}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"constructor-based terms"}, \to_{\mathcal{R}})$$

Complexity of Term Rewrite Systems

Definition

the derivation height of a term t wrt to T and $\to$

$$\mathrm{dh}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$

$$\mathrm{dh}(n, T, \to) = \max\{\mathrm{dh}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition

the derivational complexity with respect to a TRS $\mathcal{R}$

$$\mathrm{dc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"all terms"}, \to_{\mathcal{R}})$$

Definition

the runtime complexity with respect to a T

basic terms

$$\mathrm{rc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"constructor-based terms"}, \to_{\mathcal{R}})$$

Complexity of Term Rewrite Systems

Definition

the derivation height of a term t wrt to T and $\to$

$$\mathrm{dh}(t, \to) = \max\{n \mid \exists u\ t \to^n u\}$$
$$\mathrm{dh}(n, T, \to) = \max\{\mathrm{dh}(t, \to) \mid \exists t \in T \text{ and } |t| \leqslant n\}$$

Definition

the runtime complexity with respect to a TRS $\mathcal{R}$

$$\mathrm{rc}_{\mathcal{R}}(n) = \mathrm{dh}(n, \text{"constructor-based terms"}, \to_{\mathcal{R}})$$

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots$$

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ termination technique such that
2. termination of $\mathcal{R}$ is certified

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ termination technique such that
2. termination of $\mathcal{R}$ is certified

Fact

termination techniques can be used to measure the derivation height

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ termination technique such that
2. termination of $\mathcal{R}$ is certified

Fact

termination techniques can be used to measure the derivation height

Example

WIDP + POP* + argument filtering induce polynomial innermost runtime complexity

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ termination technique such that
2. termination of $\mathcal{R}$ is certified

Fact

termination techniques can be used to measure the derivation height

Exan

WIDP: restriction of DP for innermost rewriting

WIDP + POP* + argument filtering induce polynomial innermost runtime complexity

How To Analyse Complexity

$$t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} t_3 \to_{\mathcal{R}} \dots \to_{\mathcal{R}} t_n$$

consider

1. $\exists$ termination technique such that
2. termination of $\mathcal{R}$ is certified

Fact

termination techniques can be used to measure the derivation height

Exan

POP*: restriction of MPO, ¬ restriction of EPO*

WIDP + POP* + argument filtering induce polynomial innermost runtime complexity

Termination Analysis of Java Programs

Example

```
public class Int {
int val;
public static void count(Int orig, Int limit)
{
  if (orig == null || limit == null) {return;}

  Int copy = orig;

  while (orig.val < limit.val) {copy.val++;}
}}
```

Termination Analysis of Java Programs

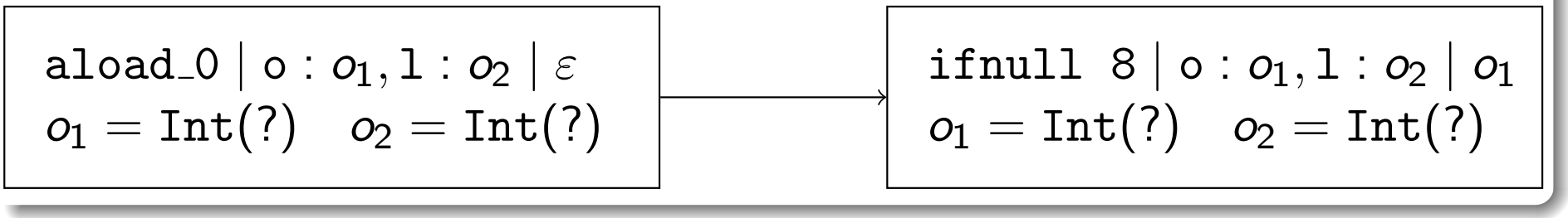
Example

```
public class Int {
int val;
public static void count(Int orig, Int limit)
{
  if (orig == null || limit == null) {return;}

  Int copy = orig;

  while (orig.val < limit.val) {copy.val++;}
}}
```

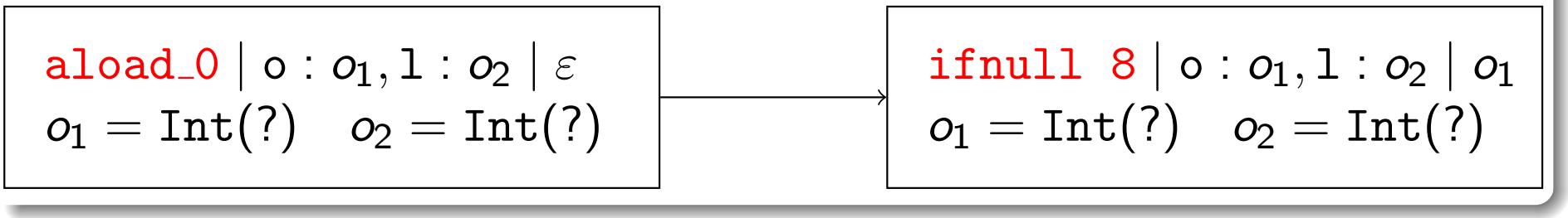
Transformation to Java Bytecode

```
$javap -c Int > Int.bc
```

Transformation to Termination Graph



Transformation to Termination Graph

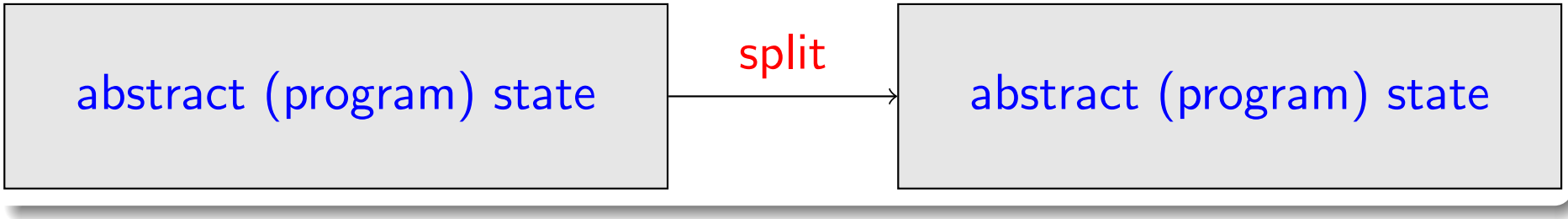


Transformation to Termination Graph

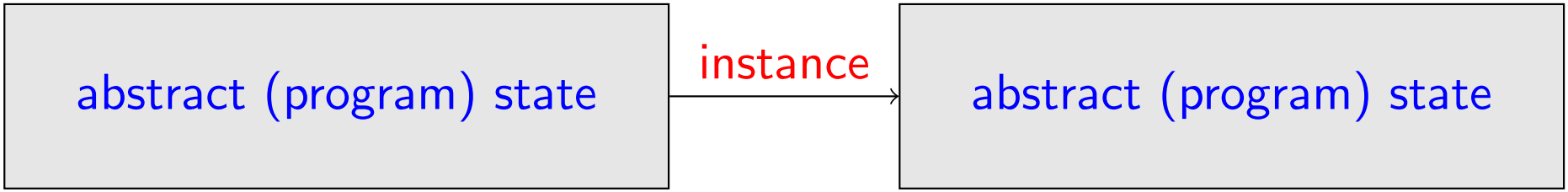
Transformation to Termination Graph

Transformation to Termination Graph

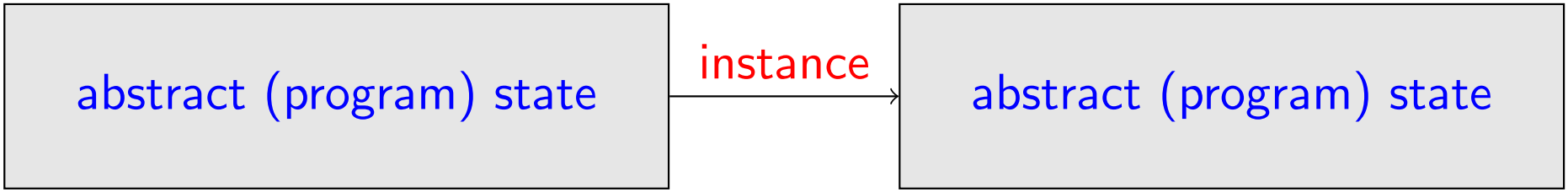
Transformation to Termination Graph



Transformation to Termination Graph

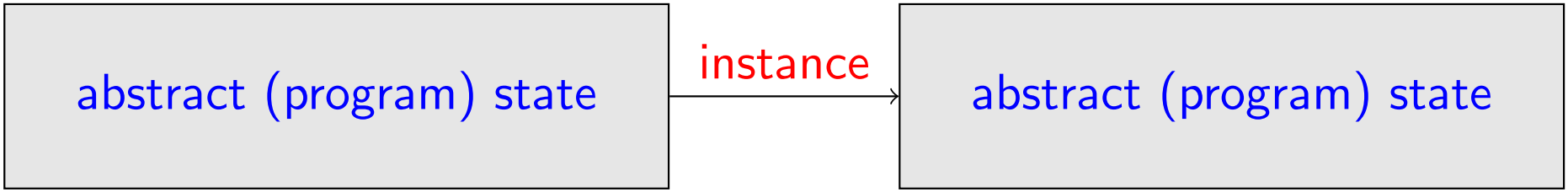


Transformation to Termination Graph



Transformation to TRS

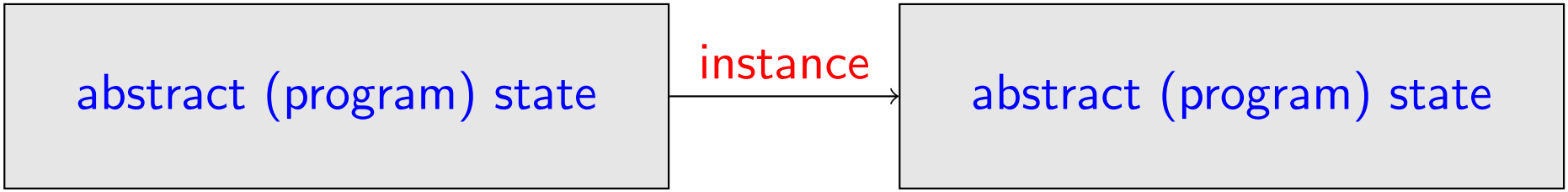
Transformation to Termination Graph



Transformation to TRS

$$\mathsf{f}_A(\mathsf{ts}(A)) \longrightarrow \mathsf{f}_B(\mathsf{ts}(B))$$

Transformation to Termination Graph



Transformation to TRS

$$\mathsf{f}_A(\mathsf{ts}(A)) \longrightarrow \mathsf{f}_B(\mathsf{ts}(B))$$

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x, y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x + 1, y)$$

Challenge ①

Question

is the runtime of any JBC program faithfully represented by the derivation height wrt its TRS representation?

Challenge ①

Question

is the runtime of any JBC program faithfully represented by the derivation height wrt its TRS representation?

Theorem (Otto et al., 2010)

if the ITRS corresponding to a termination graph is terminating, then the termination graph is terminating as well

Challenge ①

Question

is the runtime of any JBC program faithfully represented by the derivation height wrt its TRS representation?

Theorem (Otto et al., 2010)

if the ITRS corresponding to a termination graph is terminating, then the termination graph is terminating as well

Corollary

if the ITRS corresponding to a termination graph is terminating, then the original JBC program is terminating for all concrete states t that are instances of some abstract state s in the graph

Analysis of the Proof of Theorem

Analysis of the Proof of Theorem

Lemma

1. let s, t be two states in a termination graph connected by an evaluation edge

Analysis of the Proof of Theorem

Lemma

1. let s, t be two states in a termination graph connected by an evaluation edge
2. let s' be a concrete instance of s and t' be obtained by on JBC evaluation

Analysis of the Proof of Theorem

Lemma

1. let s, t be two states in a termination graph connected by an evaluation edge
2. let s' be a concrete instance of s and t' be obtained by on JBC evaluation

then

$$\mathrm{f}_s(\text{"term representation of } s'\text{"}) \rightarrow^+ \mathrm{f}_t(\text{"term representation of } t'\text{"})$$

Analysis of the Proof of Theorem

Lemma

1. let s, t be two states in a termination graph connected by an evaluation edge
2. let s' be a concrete instance of s and t' be obtained by on JBC evaluation

then

$$\mathsf{f}_s(\text{"term representation of } s'\text{"}) \rightarrow^+ \mathsf{f}_t(\text{"term representation of } t'\text{"})$$

Assessment

covers all steps in the transformation graph, hence all JBC evaluation steps lead to at least one rewrite step

Analysis of the Proof of Theorem

Lemma

1. let s, t be two states in a termination graph connected by an evaluation edge
2. let s' be a concrete instance of s and t' be obtained by on JBC evaluation

then

$$\mathsf{f}_s(\text{"term representation of } s'\text{"}) \rightarrow^+ \mathsf{f}_t(\text{"term representation of } t'\text{"})$$

Assessment

covers all steps in the transformation graph, hence all JBC evaluation steps lead to at least one rewrite step

Answer to Challenge ①

Yes

Challenge ②

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Challenge ②

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Fact

- *transformation in Otto et al. restricts to "tree-shaped" objects*
- *the JBC class* `Flatten` *doesn't*

Challenge ②

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Fact

- *transformation in Otto et al. restricts to "tree-shaped" objects*
- *the JBC class `Flatten` doesn't*

Example

```
public class Tree {
  int value;
  Tree left;
  Tree right;
  }
```

Challenge ②

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt its TRS representation?

Fact

- *transformation in Otto et al. restricts to "tree-shaped" objects*
- *the JBC class `Flatten` doesn't*

Example

```
public class Tree {
  int value;
  Tree left;
  Tree right;
  }
```

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` does not use the DAG structure

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` does not use the DAG structure

Fact

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` does not use the DAG structure

Fact

- *the number of steps in `flatten` is the same*

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` does not use the DAG structure

Fact

- *the number of steps in `flatten` is the same*
- *the runtime complexity is exponential in the size of the input*

`flatten` revisited

```
// snip
cur = new TreeList();
cur.value = tree.left;
cur.next = oldCur;
oldCur.value = tree.right;
```

`flatten` does not use the DAG structure

Fact

- *the number of steps in `flatten` is the same*
- *the runtime complexity is exponential in the size of the input*

solution: assume tree-shapes

Challenge ③

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Challenge ③

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt $\mathcal{R}_{\mathsf{flatten}}$, i.e, are the refinements complexity preserving?

Fact

the following refinements are used to shrink the rules

1. *merging of rules, for examples, rewriting of right-hand sides*
2. *create only rules for cycles in the termination graph*

Challenge ③

Question

is the runtime complexity of any JBC program faithfully represented by the runtime complexity wrt $\mathcal{R}_{\text{flatten}}$, i.e, are the refinements complexity preserving?

Fact

the following refinements are used to shrink the rules

1. *merging of rules, for examples, rewriting of right-hand sides*
2. *create only rules for cycles in the termination graph*

Answer to Challenge ③

Yes, merging of rules is not problematic and the cycles contain all necessary information

Challenge ④

Question

is the analysis extendable to ITRSs with fresh variables?

Challenge ④

Question

is the analysis extendable to ITRSs with fresh variables?

(Simplified) Definition

a integer TRS (ITRS) $\mathcal{R}$ is a TRS over $\mathcal{F} \cup \mathcal{G}$ such that

1. $\mathcal{G} = \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z} \cup \mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$, where
   - $\mathcal{F}_{\mathsf{arith}} = \{+, -, \times\}$
   - $\mathcal{F}_{\mathsf{rel}} = \{>, \geqslant, <, \leqslant\}$
   - $\mathcal{F}_{\mathsf{bool}} = \{\wedge, \rightarrow\}$

Challenge ④

Question

is the analysis extendable to ITRSs with fresh variables?

(Simplified) Definition

a integer TRS (ITRS) $\mathcal{R}$ is a TRS over $\mathcal{F} \cup \mathcal{G}$ such that

1. $\mathcal{G} = \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z} \cup \mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$, where
   - $\mathcal{F}_{\mathsf{arith}} = \{+, -, \times\}$
   - $\mathcal{F}_{\mathsf{rel}} = \{>, \geqslant, <, \leqslant\}$
   - $\mathcal{F}_{\mathsf{bool}} = \{\wedge, \rightarrow\}$
2. $\forall\ (l \rightarrow r) \in \mathcal{R},\ l \in \mathcal{T}(\mathcal{F} \cup \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}, \mathcal{V})$ and $l \notin \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}$)

Challenge ④

Question

is the analysis extendable to ITRSs with fresh variables?

(Simplified) Definition

a integer TRS (ITRS) $\mathcal{R}$ is a TRS over $\mathcal{F} \cup \mathcal{G}$ such that

1. $\mathcal{G} = \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z} \cup \mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$, where
   - $\mathcal{F}_{\mathsf{arith}} = \{+, -, \times\}$
   - $\mathcal{F}_{\mathsf{rel}} = \{>, \geqslant, <, \leqslant\}$
   - $\mathcal{F}_{\mathsf{bool}} = \{\wedge, \rightarrow\}$
2. $\forall\ (l \rightarrow r) \in \mathcal{R}$, $l \in \mathcal{T}(\mathcal{F} \cup \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}, \mathcal{V})$ and $l \notin \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}$)
3. the rewrite relation wrt a ITRS $\mathcal{R}$ is defined as

$$s \xrightarrow{\mathsf{itrs}}_{\mathcal{R}} t \iff s \xrightarrow{\mathsf{i}}_{\mathcal{R} \cup \mathcal{PD}} t$$

where $\mathcal{PD}$ collects all predefined rules for $\mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$

Challenge ④

Question

is the analysis extendable to ITRSs with fresh variables?

(Simplified) Definition

a integer TRS (ITRS) $\mathcal{R}$ is a TRS over $\mathcal{F} \cup \mathcal{G}$ such that

1. $\mathcal{G} = \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z} \cup \mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$, where
   - $\mathcal{F}_{\mathsf{arith}} = \{+, -, \times\}$
   - $\mathcal{F}_{\mathsf{rel}} = \{>, \geqslant, <, \leqslant\}$
   - $\mathcal{F}_{\mathsf{bool}} = \{\wedge, \rightarrow\}$
2. $\forall\ (l \rightarrow r) \in \mathcal{R},\ l \in \mathcal{T}(\mathcal{F} \cup \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}, \mathcal{V})$ and $l \notin \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}$)
3. the rewrite relation wrt a ITRS $\mathcal{R}$ is defined as

$$s \xrightarrow{\mathsf{itrs}}_{\mathcal{R}} t \iff s \xrightarrow{\mathsf{i}}_{\mathcal{R} \cup \mathcal{PD}} t$$

where $\mathcal{PD}$ collects all predefined rules for $\mathcal{F}_{\mathsf{arith}} \cup \mathcal{F}_{\mathsf{rel}} \cup \mathcal{F}_{\mathsf{bool}}$

Example

```
public class Exp{
  public static int exp (int x) {
    int y; int result = 1;
    while (x != 0) {
      y = result; result = 0;
      while (y != 0) {
        result += 2;
        y--;
      } x--;
    } return result;
  }
}
```

Example

```
public class Exp{
  public static int exp (int x) {
    int y; int result = 1;
    while (x != 0) {
      y = result; result = 0;
      while (y != 0) {
        result += 2;
        y--;
      } x--;
    } return result;
  }
}
```

generates the following ITRS $\mathcal{R}_{\text{while}}$

$$\mathsf{f}(x,y) \to \mathsf{fif}(x{>}0,x,y) \qquad \mathsf{fif}(\mathsf{true},x,y) \to \mathsf{g}(x,y,0)$$
$$\mathsf{g}(x,y,z) \to \mathsf{gif}((y{>}0) \wedge (z{+}2{>}0,x,y,z) \qquad \mathsf{g}(x,0,y) \to \mathsf{f}(x{-}1,y)$$
$$\mathsf{g}(\mathsf{true},x,y,z) \to \mathsf{g}(x,y{-}1,x{+}2)$$

Example

```
public class Exp{
  public static int exp (int x) {
    int y; int result = 1;
    while (x != 0) {
      y = result; result = 0;
      while (y != 0) {
        result += 2;
        y--;
      } x--;
    } return result;
  }
}
```

generates the following ITRS $\mathcal{R}_{\text{while}}$

$$\mathsf{f}(x,y) \to \mathsf{fif}(x{>}0,x,y) \qquad \mathsf{fif}(\mathsf{true},x,y) \to \mathsf{g}(x,y,0)$$
$$\mathsf{g}(x,y,z) \to \mathsf{gif}((y{>}0) \wedge (z{+}2{>}0,x,y,z) \qquad \mathsf{g}(x,0,y) \to \mathsf{f}(x{-}1,y)$$
$$\mathsf{g}(\mathsf{true},x,y,z) \to \mathsf{g}(x,y{-}1,x{+}2)$$

Example

```
public class Exp{
  public static int exp (int x) {
    int y; int result = 1;
    while (x != 0) {
      y = result; result = 0;
      while (y != 0) {
        result += 2;
        y--;
      } x--;
    } return result;
  }
}
```

generates the following ITRS $\mathcal{R}_{\text{while}}$

$$\mathsf{f}(x,y) \to \mathsf{fif}(x{>}0,x,y) \qquad \mathsf{fif}(\mathsf{true},x,y) \to \mathsf{g}(x,y,0)$$
$$\mathsf{g}(x,y,z) \to \mathsf{gif}((y{>}0) \wedge (z{+}2{>}0,x,y,z) \qquad \mathsf{g}(x,0,y) \to \mathsf{f}(x{-}1,y)$$
$$\mathsf{g}(\mathsf{true},x,y,z) \to \mathsf{g}(x,y{-}1,x{+}2)$$


C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

termination of $\mathcal{R}_{\mathsf{while}}$ *is (automatically) shown by the (I)DP framework employing*

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

termination of $\mathcal{R}_{\text{while}}$ is (automatically) shown by the (I)DP framework employing

1. *dependency graph processor*

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

termination of $\mathcal{R}_{\text{while}}$ is (automatically) shown by the (I)DP framework employing

1. *dependency graph processor*
2. *reduction pair $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ such that*
   - *$\mathcal{A}$ polynomial interpretations in $\mathbb{Z}$*
   - *$+_{\mathcal{A}}$, $-_{\mathcal{A}}$, $\times_{\mathcal{A}}$ are the natural interpretations*
   - *further sanity conditions*

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

termination of $\mathcal{R}_{\text{while}}$ is (automatically) shown by the (I)DP framework employing

1. *dependency graph processor*
2. *reduction pair $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ such that*
   - *$\mathcal{A}$ polynomial interpretations in $\mathbb{Z}$*
   - *$+_{\mathcal{A}}$, $-_{\mathcal{A}}$, $\times_{\mathcal{A}}$ are the natural interpretations*
   - *further sanity conditions*
3. *usable rules*

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

termination of $\mathcal{R}_{\text{while}}$ is (automatically) shown by the (I)DP framework employing

1. *dependency graph processor*
2. *reduction pair $(\geqslant_{\mathcal{A}}, >_{\mathcal{A}})$ such that*
   - *$\mathcal{A}$ polynomial interpretations in $\mathbb{Z}$*
   - *$+_{\mathcal{A}}$, $-_{\mathcal{A}}$, $\times_{\mathcal{A}}$ are the natural interpretations*
   - *further sanity conditions*
3. *usable rules*
4. *bounded increase*

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

polynomial complexity of $\mathcal{R}_{\text{count}}$ is shown by the WIDP method employing

1. *weak dependency graphs*
2. *reduction pair $(=_{\mathcal{A}}, >_{\mathcal{A}})$ such that*
   - *$\mathcal{A}$ polynomial interpretations in $\mathbb{Z}$*
   - *$+_{\mathcal{A}}$, $-_{\mathcal{A}}$, $\times_{\mathcal{A}}$ are the natural interpretations*
   - *further sanity conditions*
3. *usable rules*
4. ~~*bounded increase*~~

C. Fuhs, J. Giesl, M. Plücker, P. Schneider-Kamp, and S. Falke.
Proving Termination of Integer Term Rewriting.
In *Proc. of 20th RTA*, pages 32–47, 2009.

Fact

polynomial complexity of $\mathcal{R}_{\text{count}}$ is shown by the WIDP method employing

1. *weak dependency graphs*
2. *reduction pair $(=_{\mathcal{A}}, >_{\mathcal{A}})$ such that*
   - *$\mathcal{A}$ polynomial interpretations in $\mathbb{Z}$*
   - *$+_{\mathcal{A}}$, $-_{\mathcal{A}}$, $\times_{\mathcal{A}}$ are the natural interpretations*
   - *further sanity conditions*
3. *usable rules*
4. ~~*bounded increase*~~

Answer to Challenge ④

Yes

Main Result

Main Result

"Lemma"

$\forall$ termination graph $\mathcal{G}$, the length of any path starting in state s is bounded by $\mathsf{dh}(t, \xrightarrow{\text{itrs}}_{\mathcal{R}})$*, where $\mathcal{R}$, t are the ITRS, term representation respectively*

Main Result

"Lemma"

$\forall$ *termination graph* $\mathcal{G}$*, the length of any path starting in state* s *is bounded by* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$*, where* $\mathcal{R}$*,* t *are the ITRS, term representation respectively*

"Lemma"

$\forall$ *ITRS* $\mathcal{R}$*,* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, if termination of* $\mathcal{R}$ *can be shown by the WIDP method in conjunction with (safe) reduction pairs based on max-polynomials*

Main Result

"Lemma"

$\forall$ *termination graph* $\mathcal{G}$*, the length of any path starting in state* s *is bounded by* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$*, where* $\mathcal{R}$*,* t *are the ITRS, term representation respectively*

"Lemma"

$\forall$ *ITRS* $\mathcal{R}$*,* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, if termination of* $\mathcal{R}$ *can be shown by the WIDP method in conjunction with (safe) reduction pairs based on max-polynomials*

"Theorem"

$\forall$ *terminating JBC programs* P*, if* $\forall$ t $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, then* P *runs in polynomial time*

Main Result

"Lemma"

$\forall$ *termination graph* $\mathcal{G}$*, the length of any path starting in state* s *is bounded by* $\mathsf{dh}(t, \xrightarrow{\text{itrs}}_{\mathcal{R}})$*, where* $\mathcal{R}$*,* t *are the ITRS, term representation respectively*

"Lemma"

$\forall$ *ITRS* $\mathcal{R}$*,* $\mathsf{dh}(t, \xrightarrow{\text{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, if termination of* $\mathcal{R}$ *can be shown by the WIDP method in conjunction with (safe) reduction pairs based on max-polynomials*

"Theo

calls to built-in functions are constant

$\forall$ *terminating JBC programs* P*, if* $\forall\, t$ $\mathsf{dh}(t, \xrightarrow{\text{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, then* P *runs in polynomial time*

Main Result

Lemma

$\forall$ *termination graph* $\mathcal{G}$*, the length of any path starting in state* s *is bounded by* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$*, where* $\mathcal{R}$*,* t *are the ITRS, term representation respectively*

Lemma

$\forall$ *ITRS* $\mathcal{R}$*,* $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, if termination of* $\mathcal{R}$ *can be shown by the WIDP method in conjunction with (safe) reduction pairs based on max-polynomials*

Theorem

$\forall$ *terminating JBC programs* P*, if* $\forall$ t $\mathsf{dh}(t, \xrightarrow{\mathsf{itrs}}_{\mathcal{R}})$ *is polynomial in* $|t|$*, then* P *runs in polynomial time*

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x, y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x + 1, y)$$

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x, y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x + 1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*
4. $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq \gtrsim$ *and* $\mathsf{WIDP}(\mathcal{R}) \subseteq \succ$

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*
4. $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq \gtrsim$ *and* $\mathsf{WIDP}(\mathcal{R}) \subseteq \succ$

then $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}$ *is polynomial*

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*
4. $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq \gtrsim$ *and* $\mathsf{WIDP}(\mathcal{R}) \subseteq \succ$

then $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}$ *is polynomial*

but $\mathsf{WIDP}(\mathcal{R}_{\mathsf{count}})$ are duplicating ...

Really?

Example

consider ITRS $\mathcal{R}_{\mathsf{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*
4. $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq \gtrsim$ *and* $\mathsf{WIDP}(\mathcal{R}) \subseteq \succ$

then $\mathsf{rc}^{\mathsf{i}}_{\mathcal{R}}$ *is polynomial*

but $\mathsf{WIDP}(\mathcal{R}_{\mathsf{count}})$ are duplicating ... solved: topic for another talk ☺

Really, but not yet automated

Example

consider ITRS $\mathcal{R}_{\text{count}}$:

$$\mathsf{f}(x,y) \to \mathsf{fif}(x < y, x, y) \qquad \mathsf{fif}(\mathsf{true}, x, y) \to \mathsf{f}(x+1, y)$$

Theorem (Hirokawa,M, 2008)

$\forall$ *TRS* $\mathcal{R}$, *assume*

1. $\exists$ *SLI* $\mathcal{A}$, *such that* $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq >_{\mathcal{A}}$
2. $\exists$ *a reduction pair* $(\gtrsim, \succ)$
3. $\mathsf{WIDP}(\mathcal{R})$ *is non-duplicating*
4. $\mathcal{U}(\mathsf{WIDP}(\mathcal{R})) \subseteq \gtrsim$ *and* $\mathsf{WIDP}(\mathcal{R}) \subseteq \succ$

then $\mathrm{rc}^{\mathrm{i}}_{\mathcal{R}}$ *is polynomial*

but $\mathsf{WIDP}(\mathcal{R}_{\text{count}})$ are duplicating ... solved: topic for another talk ☺

Thank You for Your Attention!