



# **Termination and Confluence: Remembering Hans Zantema**

Aart Middeldorp

University of Innsbruck







## ▶ Dieter Hofbauer




- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann





- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann
- ▶  **dblp**  
computer science bibliography




- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann
- ▶  **dblp**  
computer science bibliography
- ▶ Hans Zantema




- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann
- ▶  **dblp**  
computer science bibliography
- ▶ Hans Zantema 



- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann
- ▶  **dblp**  
computer science bibliography
- ▶ Hans Zantema





- ▶ Dieter Hofbauer
- ▶ Naoki Nishida
- ▶ Johannes Waldmann
- ▶  **dblp**  
computer science bibliography
- ▶ Hans Zantema



# Outline

- 1. Hans Zantema**
- 2. WST & IWC**
- 3. Hans Zantema**
- 4. Workshops and Competitions**
- 5. Hans Zantema**





1956 — 2025

# Numbers

► publications venues:

- ① RTA 18
- ② I&C 7
- ③ AAECC 6
- ④ EPTCS 5

# Numbers

## ► publications venues:

- ① RTA 18
- ② I&C 7
- ③ AAECC 6
- ④ EPTCS 5

## ► coauthors:

- ① Jörg Endrullis 11
- ① Alfons Geser 11
- ① Henk Don 11
- ④ Aart Middeldorp 7
- ④ Johannes Waldmann 7

# Numbers

## ► publications venues:

- ① RTA 18
- ② I&C 7
- ③ AAECC 6
- ④ EPTCS 5

## ► coauthors:

- ① Alfons Geser 11
- ② Jörg Endrullis 7
- ② Aart Middeldorp 7
- ② Johannes Waldmann 7
- ⑤ Henk Don 6

► organization:

**1997** WST

► organization:

**1997** WST

**2013** RTA

► organization:

**1997** WST

**2013** RTA

**2017** ISR

► organization:

**1997** WST

**2013** RTA

**2017** ISR

► PhD students:

**1995** Maria Ferreira

**1997** Thomas Arts

**2005** Olga Tveretina

**2008** Adam Koprowski

**2011** Matthias Raffelsieper



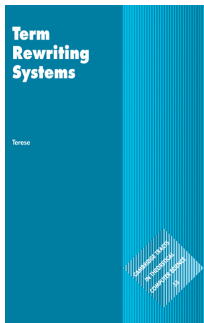
► organization:

**1997** WST

**2013** RTA

**2017** ISR

► books / book chapters:



2003

► PhD students:

**1995** Maria Ferreira

**1997** Thomas Arts

**2005** Olga Tveretina

**2008** Adam Koprowski

**2011** Matthias Raffelsieper

► organization:

**1997** WST

**2013** RTA

**2017** ISR

► books / book chapters:

► PhD students:

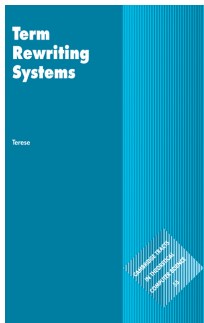
**1995** Maria Ferreira

**1997** Thomas Arts

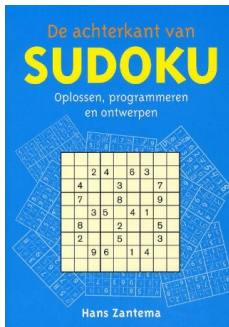
**2005** Olga Tveretina

**2008** Adam Koprowski

**2011** Matthias Raffelsieper



2003



2007

► organization:

**1997** WST

**2013** RTA

**2017** ISR

► books / book chapters:

► PhD students:

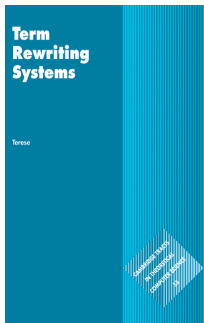
**1995** Maria Ferreira

**1997** Thomas Arts

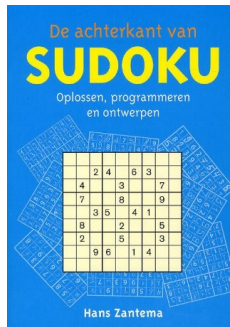
**2005** Olga Tveretina

**2008** Adam Koprowski

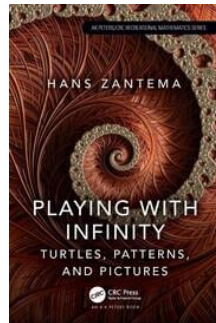
**2011** Matthias Raffelsieper



2003



2007



2024

► organization:

**1997** WST

**2013** RTA

**2017** ISR

► books / book chapters:

► PhD students:

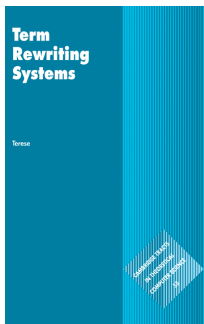
**1995** Maria Ferreira

**1997** Thomas Arts

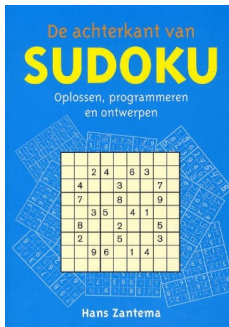
**2005** Olga Tveretina

**2008** Adam Koprowski

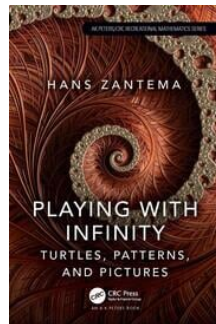
**2011** Matthias Raffelsieper



2003



2007



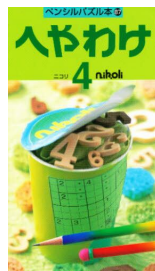
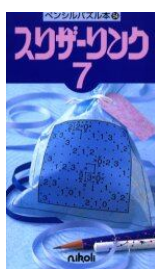
2024

*Alweer een hele tijd geleden, het zal ergens in de eerste helft van de jaren negentig geweest zijn, maakte ik met een stel collega's een flinke wandeling door de duinen van Egmond en Schoorl. Een van die collega's zat sinds een paar jaar in Japan, maar het was een soort traditie geworden om een flinke wandeling te maken als hij weer eens terug kwam in Nederland. Aan het einde van die wandeling haalde hij ineens een stapel cadeautjes tevoorschijn: Japanse puzzelboekjes. Onze eerste reactie was: wat moeten wij daar nou mee, wij lezen toch geen Japans? Omgekeerd zal een Japanner toch ook niet echt blij te maken zijn met een boekje vol met Nederlandse kruiswoordraadsels.*

*Alweer een hele tijd geleden, het zal ergens in de eerste helft van de jaren negentig geweest zijn, maakte ik met een stel collega's een flinke wandeling door de duinen van Egmond en Schoorl. Een van die collega's zat sinds een paar jaar in Japan, maar het was een soort traditie geworden om een flinke wandeling te maken als hij weer eens terug kwam in Nederland. Aan het einde van die wandeling haalde hij ineens een stapel cadeautjes tevoorschijn: Japanse puzzelboekjes. Onze eerste reactie was: wat moeten wij daar nou mee, wij lezen toch geen Japans? Omgekeerd zal een Japanner toch ook niet echt blij te maken zijn met een boekje vol met Nederlandse kruiswoordraadsels.*

*A long time ago, probably sometime in the early 1990s, I took a long walk through the dunes of Egmond and Schoorl with a group of colleagues. One of those colleagues had been in Japan for a few years, but it had become something of a tradition to take a long walk whenever he returned to the Netherlands. At the end of that walk, he suddenly pulled out a stack of presents: Japanese puzzle books. Our initial reaction was: what are we supposed to do with that? We don't read Japanese anyway. Conversely, a Japanese person wouldn't be exactly thrilled with a book full of Dutch crosswords either.*







ISR 2010



[www.phil.uu.nl/isr2010/](http://www.phil.uu.nl/isr2010/)

5th International School on Rewriting  
July 3 – 8, 2010, Utrecht, The Netherlands

Under the auspices of  
IFIP WG 1.6 – Term Rewriting  
Organised by Utrecht Summerschool  
<http://www.utrechtsummerschool.nl>

#### Organisers

- Vincent van Oostrom, Utrecht
- Roel de Vrijer, Amsterdam

#### Lecturers

- Aart Middeldorp, Innsbruck
- Femke van Raamsdonk, Amsterdam
- Ashish Tiwari, Menlo Park
- Ralf Treinen, Paris
- Peter Schneider-Kamp, Odense
- Hans Zantema, Eindhoven
- Adam Koprowski, Paris
- Georg Moser, Innsbruck
- Dimitri Hendriks, Amsterdam
- Jörg Endrullis, Amsterdam
- Clemens Grabmayer, Utrecht



summer school  
UTRECHT

Hochschule für Technik, Wirtschaft und Kultur Leipzig  
Leipzig University of Applied Sciences



## 8th International School on Rewriting

August 10 – 14, 2015 — Leipzig, Germany



#### Basic Track

- › Aart Middeldorp and Sarah Winkler:  
Rewriting: Basic Course for Beginners

#### Advanced Track

- › Santiago Escobar: Term Rewriting applied to Cryptographic Protocol Analysis: the Maude-NPA tool
- › Alfons Gser: Proving Abstract Rewriting Properties with PVS
- › Makoto Hamana: Algebraic Semantics of Higher-Order Abstract Syntax and Second-Order Rewriting

- › Georg Moser: Termination and Complexity
- › Detlef Plump: Rule-based Graph Programming
- › David Sabel and Manfred Schmidt-Schauss: Rewriting Techniques for Correctness of Program Transformations
- › Femke van Raamsdonk: Higher-Order Rewriting
- › Hans Zantema: SAT/SMT encodings for rewrite problems

#### Organizing Committee

- › Alfons Gser, Christine Klöden,  
Johannes Waldmann

Registration deadline: July 1, 2015

FAKULTÄT  
INFORMATIK, MATHEMATIK  
UND NATURWISSENSCHAFTEN

[www.tmm.htwk-leipzig.de/ISR2015/](http://www.tmm.htwk-leipzig.de/ISR2015/)



IFIP Working Group 1.6  
Term Rewriting



Photo: Leipzig, 2014

# 5th International School on Rewriting

## 3 – 8 July 2010, Drift 21, Utrecht, Netherlands

### SAT Solving for Term Rewriting

#### Lecturer

[Hans Zantema](#) ([Technische Universiteit Eindhoven](#) and [Radbout Universiteit Nijmegen](#), The Netherlands)

#### Introduction

(advanced track, 2 sessions, mixed theoretical and practical)

SATisfiability is the following problem: given a propositional formula over a set of Boolean variables, decide whether it is satisfiable, that is, Boolean values can be assigned to the variables such that the formula yields true. Current SAT solvers easily solve instances over thousands of variables.

Several problems in term rewriting are essentially constraint problems that can be expressed in SAT solving. For instance, proving termination of a TRS by recursive path order corresponds to finding a precedence and status function such that a number of conditions hold. Rather than writing an algorithm for searching for a precedence and status function, one can express the requirements as a SAT problem, call a SAT solver, and then extract the found precedence and status function from the satisfying assignment found by the SAT solver. The latter approach turns out to be much more efficient. Similar observations hold for other techniques for proving termination, like finding argument filters and polynomial interpretations.

All current termination solvers like AProVE and TTT2 heavily use SAT solving. SAT solving also applies for other problems, like proving confluence or checking convertibility. More recent developments make use of SMT solving (satisfiability modulo theories), being an extension of SAT solving in which, for instance, one can express linear inequalities on integers directly, instead of encoding these inequalities as a SAT formula on the bits of the binary encoding of the numbers.

# SAT/SMT encodings for rewrite problems

Hans Zantema, Department of Computer Science, TU Eindhoven, The Netherlands

For automatically checking whether a rewrite system satisfies some property, applying SAT solving has shown up to be extremely useful. The presentation will be split up into the following parts. In the first part it is shown how various constraint problems independent from rewriting can be expressed and solved by SAT/SMT solving, and some basic theory is presented. Next there is a practical slot in which students solve exercise problems, using SAT/SMT tools on their laptop. In the last part it will be shown how this approach can be applied for rewriting problems, in particular, proving termination and non-termination of rewriting.

[Hans Zantema](#) is associate professor at Eindhoven University of Technology and part time full professor at Radboud University in Nijmegen. He did his PhD in algebraic number theory in 1983. Since 1990 one of his main research interests is in rewriting; first with a focus on termination, later on with a focus on both applying SAT solving and on infinite data structures.

## ► Termination of Rewriting: Interpretation and Type Elimination

243

JSC 1994

*J. Symbolic Computation* (1994) 17, 23–50

### Termination of term rewriting: interpretation and type elimination

H. Zantema

*Utrecht University, Department of Computer Science*

*P.O. box 80.089, 3508 TB Utrecht, The Netherlands*

*e-mail: hansz@cs.ruu.nl*

*(Received 25 March 1993)*

We investigate proving termination of term rewriting systems by interpretation of terms in a well-founded monotone algebra. The well-known polynomial interpretations can be considered as a particular case in this framework. A classification of types of termination, including simple termination, is proposed based on properties in the semantic level. A transformation on term rewriting systems eliminating distributive rules is introduced. Using this distribution elimination a new termination proof of the system SUBST of Hardin and Laville (1986) is given. This system describes explicit substitution in  $\lambda$ -calculus.

Another tool for proving termination is based on introduction and removal of type restrictions. A property of many-sorted term rewriting systems is called persistent if it is not affected by removing the corresponding typing restriction. Persistence turns out to be a generalization of distributivity but is more powerful for both

## ► Termination of Rewriting: Interpretation and Type Elimination

243

JSC 1994

*J. Symbolic Computation* (1994) 17, 23–50

### Termination of term rewriting: interpretation and type elimination

H. Zantema

*Utrecht University, Department of Computer Science*

*P.O. box 80.089, 3508 TB Utrecht, The Netherlands*

*e-mail: hansz@cs.ruu.nl*

*(Received 25 March 1993)*

We investigate proving termination of term rewriting systems by interpretation of terms in a well-founded monotone algebra. The well-known polynomial interpretations can be considered as a particular case in this framework. A classification of types of termination, including simple termination, is proposed based on properties in the semantic level. A transformation on term rewriting systems eliminating distributive rules is introduced. Using this distribution elimination a new termination proof of the system SUBST of Hardin and Laville (1986) is given. This system describes explicit substitution in  $\lambda$ -calculus.

Another tool for proving termination is based on introduction and removal of type restrictions. A property of many-sorted term rewriting systems is called persistent if it is not affected by removing the corresponding typing restriction. Persistence turns out to be a generalization of distributivity but is more powerful for both

## Definitions

given non-empty set  $\mathcal{S}$  of **sorts**

given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  **$\mathcal{S}$ -sorted** if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with **sort declaration**  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$

## Definitions

given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  $\mathcal{S}$ -sorted if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$
- ▶ set of variables  $\mathcal{V}$  is partitioned into countably infinite sets  $\mathcal{V}_\alpha$  for every sort  $\alpha \in \mathcal{S}$



given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  $\mathcal{S}$ -sorted if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$
- ▶ set of variables  $\mathcal{V}$  is partitioned into countably infinite sets  $\mathcal{V}_\alpha$  for every sort  $\alpha \in \mathcal{S}$
- ▶ set  $\mathcal{T}_{\mathcal{S}}(\mathcal{F}, \mathcal{V})$  of **well-typed terms** is union of  $\mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  for  $\alpha \in \mathcal{S}$ 
  - ▶  $\mathcal{V}_\alpha \subseteq \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$
  - ▶  $f(t_1, \dots, t_n) \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  whenever  $f \in \mathcal{F}$  has sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  and  $t_i \in \mathcal{T}_{\alpha_i}(\mathcal{F}, \mathcal{V})$  for all  $1 \leq i \leq n$

## Definitions

given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  $\mathcal{S}$ -sorted if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$
- ▶ set of variables  $\mathcal{V}$  is partitioned into countably infinite sets  $\mathcal{V}_\alpha$  for every sort  $\alpha \in \mathcal{S}$
- ▶ set  $\mathcal{T}_\mathcal{S}(\mathcal{F}, \mathcal{V})$  of well-typed terms is union of  $\mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  for  $\alpha \in \mathcal{S}$ 
  - ▶  $\mathcal{V}_\alpha \subseteq \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$
  - ▶  $f(t_1, \dots, t_n) \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  whenever  $f \in \mathcal{F}$  has sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  and  $t_i \in \mathcal{T}_{\alpha_i}(\mathcal{F}, \mathcal{V})$  for all  $1 \leq i \leq n$
- ▶ if  $t \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  then  $t$  has sort  $\alpha$

given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  $\mathcal{S}$ -sorted if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$
- ▶ set of variables  $\mathcal{V}$  is partitioned into countably infinite sets  $\mathcal{V}_\alpha$  for every sort  $\alpha \in \mathcal{S}$
- ▶ set  $\mathcal{T}_\mathcal{S}(\mathcal{F}, \mathcal{V})$  of well-typed terms is union of  $\mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  for  $\alpha \in \mathcal{S}$ 
  - ▶  $\mathcal{V}_\alpha \subseteq \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$
  - ▶  $f(t_1, \dots, t_n) \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  whenever  $f \in \mathcal{F}$  has sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  and  $t_i \in \mathcal{T}_{\alpha_i}(\mathcal{F}, \mathcal{V})$  for all  $1 \leq i \leq n$
- ▶ if  $t \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  then  $t$  has sort  $\alpha$
- ▶ **many-sorted** TRS  $\mathcal{R}$  consists of  $\mathcal{S}$ -sorted signature and rewrite rules between well-typed terms of equal sort

given non-empty set  $\mathcal{S}$  of sorts

- ▶ signature  $\mathcal{F}$  is  $\mathcal{S}$ -sorted if every  $n$ -ary function symbol  $f \in \mathcal{F}$  is equipped with sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  where  $\alpha_1, \dots, \alpha_n, \alpha \in \mathcal{S}$
- ▶ set of variables  $\mathcal{V}$  is partitioned into countably infinite sets  $\mathcal{V}_\alpha$  for every sort  $\alpha \in \mathcal{S}$
- ▶ set  $\mathcal{T}_\mathcal{S}(\mathcal{F}, \mathcal{V})$  of well-typed terms is union of  $\mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  for  $\alpha \in \mathcal{S}$ 
  - ▶  $\mathcal{V}_\alpha \subseteq \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$
  - ▶  $f(t_1, \dots, t_n) \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  whenever  $f \in \mathcal{F}$  has sort declaration  $\alpha_1 \times \cdots \times \alpha_n \rightarrow \alpha$  and  $t_i \in \mathcal{T}_{\alpha_i}(\mathcal{F}, \mathcal{V})$  for all  $1 \leq i \leq n$
- ▶ if  $t \in \mathcal{T}_\alpha(\mathcal{F}, \mathcal{V})$  then  $t$  has sort  $\alpha$
- ▶ many-sorted TRS  $\mathcal{R}$  consists of  $\mathcal{S}$ -sorted signature and rewrite rules between well-typed terms of equal sort
- ▶  $\Theta(\mathcal{R})$  is TRS obtained from many-sorted TRS  $\mathcal{R}$  by forgetting sorts

## Example

### ► TRS

$$0 + y \rightarrow y$$

$$s(x) + y \rightarrow s(x + y)$$

$$\text{nil} ++ z \rightarrow z$$

$$c(x, y) ++ z \rightarrow c(x, y ++ z)$$

## Example

### ► TRS

$$\begin{aligned}0 + y &\rightarrow y \\ s(x) + y &\rightarrow s(x + y)\end{aligned}$$

$$\begin{aligned}\text{nil} ++ z &\rightarrow z \\ c(x, y) ++ z &\rightarrow c(x, y ++ z)\end{aligned}$$

### ► compatible sort declarations

$$0 : \mathbf{N} \quad s : \mathbf{N} \rightarrow \mathbf{N} \quad + : \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N} \quad \text{nil} : \mathbf{L} \quad c : \mathbf{N} \times \mathbf{L} \rightarrow \mathbf{L} \quad ++ : \mathbf{L} \times \mathbf{L} \rightarrow \mathbf{L}$$

## Example

### ► TRS

$$0 + y \rightarrow y$$

$$s(x) + y \rightarrow s(x + y)$$

$$\text{nil} ++ z \rightarrow z$$

$$c(x, y) ++ z \rightarrow c(x, y ++ z)$$

### ► compatible sort declarations

$$0 : N \quad s : N \rightarrow N \quad + : N \times N \rightarrow N \quad \text{nil} : L \quad c : N \times L \rightarrow L \quad ++ : L \times L \rightarrow L$$

### ► well-typed terms

$$s(s(0)) \quad c(s(0), c(0, \text{nil})) \quad c(s(0), \text{nil} ++ \text{nil})$$

## Example

### ► TRS

$$0 + y \rightarrow y$$

$$s(x) + y \rightarrow s(x + y)$$

$$\text{nil} ++ z \rightarrow z$$

$$c(x, y) ++ z \rightarrow c(x, y ++ z)$$

### ► compatible sort declarations

$$0 : \mathbf{N} \quad s : \mathbf{N} \rightarrow \mathbf{N} \quad + : \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N} \quad \text{nil} : \mathbf{L} \quad c : \mathbf{N} \times \mathbf{L} \rightarrow \mathbf{L} \quad ++ : \mathbf{L} \times \mathbf{L} \rightarrow \mathbf{L}$$

### ► well-typed terms

$$s(s(0)) \quad c(s(0), c(0, \text{nil})) \quad c(s(0), \text{nil} ++ \text{nil})$$

### ► ill-typed terms

$$s(s(\text{nil})) \quad c(s(0), c(0, 0)) \quad c(s(0), s(0) ++ \text{nil})$$



## Definition

property  $P$  of many-sorted TRSs is **persistent** if

$$\mathcal{R} \text{ satisfies } P \iff \Theta(\mathcal{R}) \text{ satisfies } P$$

## Definition

property  $P$  of many-sorted TRSs is persistent if

$$\mathcal{R} \text{ satisfies } P \iff \Theta(\mathcal{R}) \text{ satisfies } P$$

## Example

many-sorted TRS  $\mathcal{R}$       $f(a, b, x) \rightarrow f(x, x, x)$       $g(y, z) \rightarrow y$       $g(y, z) \rightarrow z$

## Definition

property  $P$  of many-sorted TRSs is persistent if

$$\mathcal{R} \text{ satisfies } P \iff \Theta(\mathcal{R}) \text{ satisfies } P$$

## Example

many-sorted TRS  $\mathcal{R}$      $f(a, b, x) \rightarrow f(x, x, x)$      $g(y, z) \rightarrow y$      $g(y, z) \rightarrow z$

with sorts  $\{\alpha, \beta\}$  and sort declarations

$a, b : \alpha$

$f : \alpha \times \alpha \times \alpha \rightarrow \alpha$

$g : \beta \times \beta \rightarrow \beta$

## Definition

property  $P$  of many-sorted TRSs is persistent if

$$\mathcal{R} \text{ satisfies } P \iff \Theta(\mathcal{R}) \text{ satisfies } P$$

## Example

many-sorted TRS  $\mathcal{R}$       $f(a, b, x) \rightarrow f(x, x, x)$       $g(y, z) \rightarrow y$       $g(y, z) \rightarrow z$

with sorts  $\{\alpha, \beta\}$  and sort declarations

$a, b : \alpha$

$f : \alpha \times \alpha \times \alpha \rightarrow \alpha$

$g : \beta \times \beta \rightarrow \beta$

is terminating

### Definition

property  $P$  of many-sorted TRSs is persistent if

$$\mathcal{R} \text{ satisfies } P \iff \Theta(\mathcal{R}) \text{ satisfies } P$$

many-sorted TRS  $\mathcal{R}$        $f(a, b, x) \rightarrow f(x, x, x)$        $g(y, z) \rightarrow y$        $g(y, z) \rightarrow z$   
with sorts  $\{\alpha, \beta\}$  and sort declarations

$$\mathbf{a}, \mathbf{b} : \alpha \qquad \mathbf{f} : \alpha \times \alpha \times \alpha \rightarrow \alpha \qquad \mathbf{g} : \beta \times \beta \rightarrow \beta$$

is terminating but  $\Theta(\mathcal{R})$  is not terminating:

$$f(g(a,b), g(a,b), g(a,b)) \longrightarrow f(a, g(a,b), g(a,b))$$

$\quad \quad \quad \nearrow \qquad \qquad \nwarrow$

$$\qquad \qquad \qquad f(a,b,g(a,b))$$

## Theorem

termination is persistent for TRSs that do not contain both collapsing and duplicating rules

termination is persistent for TRSs that do not contain both collapsing and duplicating rules

that it is a generalization of direct sum modularity. We proved that termination is a persistent property for a similar class of TRS's for which Rusinowitch (1987) proved termination to be modular. This result provides a new and powerful tool for proving termination of TRS's.

Proving persistence of weak normalization and weak confluence is not difficult; proving persistence of confluence seems to be feasible. For notions like simple termination as considered in the first part of the paper the notion of persistence does not make sense since it is not clear how to define simple termination for many-sorted TRS-s. As a conjecture we state that termination is a persistent property for the class of TRS's in which all variables are of the same sort.

termination is persistent for TRSs that do not contain both collapsing and duplicating rules

that it is a generalization of direct sum modularity. We proved that termination is a persistent property for a similar class of TRS's for which Rusinowitch (1987) proved termination to be modular. This result provides a new and powerful tool for proving termination of TRS's.

Proving persistence of weak normalization and weak confluence is not difficult; proving persistence of confluence seems to be feasible. For notions like simple termination as considered in the first part of the paper the notion of persistence does not make sense since it is not clear how to define simple termination for many-sorted TRS-s. As a conjecture we state that termination is a persistent property for the class of TRS's in which all variables are of the same sort.



# Solution to the Problem of Zantema on a Persistent Property of Term Rewriting Systems\*

Takahito Aoto

Department of Computer Science, Gunma University  
Tenjincho 1-5-1, Kiryuu, 376-8515, Japan

E-mail: [aoto@cs.gunma-u.ac.jp](mailto:aoto@cs.gunma-u.ac.jp)

## Abstract

A property  $P$  of term rewriting systems is persistent if for any many-sorted term rewriting system  $R$ ,  $R$  has the property  $P$  if and only if its underlying term rewriting system  $\Theta(R)$ , which results from  $R$  by omitting its sort information, has the property  $P$ . It is shown that termination is a persistent property of many-sorted term rewriting systems that contain only variables of the same sort.

## 1 Introduction

The technique of sort introduction in term rewriting has caught attention

## Example

► TRS

$$f(g(a), g(b), x) \rightarrow f(x, x, x)$$
$$g(x) \rightarrow x$$

# Solution to the Problem of Zantema on a Persistent Property of Term Rewriting Systems\*

Takahito Aoto

Department of Computer Science, Gunma University  
Tenjincho 1-5-1, Kiryuu, 376-8515, Japan

E-mail: aoto@cs.gunma-u.ac.jp

## Abstract

A property  $P$  of term rewriting systems is persistent if for any many-sorted term rewriting system  $R$ ,  $R$  has the property  $P$  if and only if its underlying term rewriting system  $\Theta(R)$ , which results from  $R$  by omitting its sort information, has the property  $P$ . It is shown that termination is a persistent property of many-sorted term rewriting systems that contain only variables of the same sort.

## 1 Introduction

The technique of sort introduction in term rewriting has caught attention

## Example

► TRS

$$f(g(a), g(b), x) \rightarrow f(x, x, x)$$

$$g(x) \rightarrow x$$

► sorts  $\{\alpha, \beta\}$  and sort declarations

$$a, b : \alpha$$

$$g : \alpha \rightarrow \alpha$$

$$f : \alpha \times \alpha \times \alpha \rightarrow \beta$$

# Solution to the Problem of Zantema on a Persistent Property of Term Rewriting Systems\*

Takahito Aoto

Department of Computer Science, Gunma University  
Tenjincho 1-5-1, Kiryuu, 376-8515, Japan

E-mail: aoto@cs.gunma-u.ac.jp

## Abstract

A property  $P$  of term rewriting systems is persistent if for any many-sorted term rewriting system  $R$ ,  $R$  has the property  $P$  if and only if its underlying term rewriting system  $\Theta(R)$ , which results from  $R$  by omitting its sort information, has the property  $P$ . It is shown that termination is a persistent property of many-sorted term rewriting systems that contain only variables of the same sort.

## 1 Introduction

The technique of sort introduction in term rewriting has caught attention

## Example

► TRS

$$f(g(a), g(b), x) \rightarrow f(x, x, x)$$

$$g(x) \rightarrow x$$

► sorts  $\{\alpha, \beta\}$  and sort declarations

$$a, b : \alpha$$

$$g : \alpha \rightarrow \alpha$$

$$f : \alpha \times \alpha \times \alpha \rightarrow \beta$$

►  $x : \alpha$

# Solution to the Problem of Zantema on a Persistent Property of Term Rewriting Systems\*

Takahito Aoto

Department of Computer Science, Gunma University  
Tenjincho 1-5-1, Kiryuu, 376-8515, Japan

E-mail: aoto@cs.gunma-u.ac.jp

## Abstract

A property  $P$  of term rewriting systems is persistent if for any many-sorted term rewriting system  $R$ ,  $R$  has the property  $P$  if and only if its underlying term rewriting system  $\Theta(R)$ , which results from  $R$  by omitting its sort information, has the property  $P$ . It is shown that termination is a persistent property of many-sorted term rewriting systems that contain only variables of the same sort.

## 1 Introduction

The technique of sort introduction in term rewriting has caught attention

## Definition

property  $P$  of ARSs is **component-closed** if

ARS  $\langle A, \rightarrow \rangle$  satisfies  $P \iff$

ARS  $\langle A, \rightarrow \cap (C_a \times C_a) \rangle$  with  $C_a = \{b \mid a \leftrightarrow^* b\}$  satisfies  $P$  for all  $a \in A$

## Definition

property  $P$  of ARSs is component-closed if

ARS  $\langle A, \rightarrow \rangle$  satisfies  $P \iff$

ARS  $\langle A, \rightarrow \cap (C_a \times C_a) \rangle$  with  $C_a = \{b \mid a \leftrightarrow^* b\}$  satisfies  $P$  for all  $a \in A$

## Remark

almost all properties of interest (SN, CR, UNC, GCR, ...) are component-closed

## Definition

property  $P$  of ARSs is component-closed if

ARS  $\langle A, \rightarrow \rangle$  satisfies  $P \iff$

ARS  $\langle A, \rightarrow \cap (C_a \times C_a) \rangle$  with  $C_a = \{b \mid a \leftrightarrow^* b\}$  satisfies  $P$  for all  $a \in A$

## Remark

almost all properties of interest (SN, CR, UNC, GCR, ...) are component-closed

## Theorem (Zantema)

every component-closed persistent property is modular

# Modularity in Many-sorted Term Rewriting Systems

Jaco van de Pol

Januar 1993

## Abstract

Many-sorted term rewriting systems (MTRS) are an extension of the formalism of term rewriting systems (TRS). The *direct sum* of TRS's is generalized to the direct sum of MTRS's. Some equivalence between “taking the direct sum” and “eliminating the sorts” is established: Component closed reduction properties which are resistant against disjoint union (*modularity*), are also resistant against sort elimination (*persistence*). The reverse has been proved earlier.



## Theorem

component-closed modular properties of many-sorted TRSs with finitely many sorts are persistent

# Modularity in Many-sorted Term Rewriting Systems

Jaco van de Pol

Januar 1993

## Abstract

Many-sorted term rewriting systems (MTRS) are an extension of the formalism of term rewriting systems (TRS). The *direct sum* of TRS's is generalized to the direct sum of MTRS's. Some equivalence between “taking the direct sum” and “eliminating the sorts” is established: Component closed reduction properties which are resistant against disjoint union (*modularity*), are also resistant against sort elimination (*persistence*). The reverse has been proved earlier.

## Theorem

component-closed modular properties of  
many-sorted TRSs with finitely many sorts  
are persistent

This thesis can be seen as a continuation of the research of H. Zantema on the persistence of properties of many-sorted TRS's. A property is called *persistent* if it is preserved by the sort-elimination-operator on many-sorted TRS's. Zantema proved that persistent properties are modular, for some class of properties (reduction properties or component closed properties). In fact we can partially answer one of the “Concluding remarks” of his article [Zan91, 628]: “It is unknown whether there are modular reduction properties that are not persistent”: For modular properties on many-sorted TRS's the answer is: No, there are none.

### 1.2 Overview and Main Results

You are reading my master thesis, which is the closure of my study in Computer Science on the “Rijksuniversiteit Utrecht”. I am grateful to Hans Zantema for being present when I needed him. It sometimes took him 7 hours a day. In the rest of this section an overview is given of the contents of this thesis.

## Modularity in Many-sorted Term Rewriting Systems

Jaco van de Pol

Januar 1993

### Abstract

Many-sorted term rewriting systems (MTRS) are an extension of the formalism of term rewriting systems (TRS). The *direct sum* of TRS's is generalized to the direct sum of MTRS's. Some equivalence between “taking the direct sum” and “eliminating the sorts” is established: Component closed reduction properties which are resistant against disjoint union (*modularity*), are also resistant against sort elimination (*persistence*). The reverse has been proved earlier.

## Theorem

component-closed modular properties of  
many-sorted TRSs with finitely many sorts  
are persistent

This thesis can be seen as a continuation of the research of H. Zantema on the persistence of properties of many-sorted TRS's. A property is called *persistent* if it is preserved by the sort-elimination-operator on many-sorted TRS's. Zantema proved that persistent properties are modular, for some class of properties (reduction properties or component closed properties). In fact we can partially answer one of the “Concluding remarks” of his article [Zan91, 628]: “It is unknown whether there are modular reduction properties that are not persistent”: For modular properties on many-sorted TRS's the answer is: No, there are none.

### 1.2 Overview and Main Results

You are reading my master thesis, which is the closure of my study in Computer Science on the “Rijksuniversiteit Utrecht”. I am grateful to Hans Zantema for being present when I needed him. It sometimes took him 7 hours a day. In the rest of this section an overview is given of the contents of this thesis.

## Modularity in Many-sorted Term Rewriting Systems

Jaco van de Pol

Januar 1993

### Abstract

Many-sorted term rewriting systems (MTRS) are an extension of the formalism of term rewriting systems (TRS). The *direct sum* of TRS's is generalized to the direct sum of MTRS's. Some equivalence between “taking the direct sum” and “eliminating the sorts” is established: Component closed reduction properties which are resistant against disjoint union (*modularity*), are also resistant against sort elimination (*persistence*). The reverse has been proved earlier.

## Type Introduction for Equational Rewriting

Hitoshi Ohsaki and Aart Middeldorp

Institute of Information Sciences and Electronics  
University of Tsukuba  
Tsukuba 305, Japan  
{hitoshi,ami}@score.is.tsukuba.ac.jp

**Abstract.** Type introduction is a useful technique for simplifying the task of proving properties of rewrite systems by restricting the set of terms that have to be considered to the well-typed terms according to any many-sorted type discipline which is compatible with the rewrite system under consideration. A property of rewrite systems for which type introduction is correct is called persistent. Zantema showed that termination is a persistent property of non-collapsing rewrite systems and non-duplicating rewrite systems. We extend his result to the more complicated case of equational rewriting. As a simple application we prove the undecidability of AC-termination for terminating rewrite systems. We also present sufficient conditions for the persistence of acyclicity and non-loopingness, two properties which guarantee the absence of certain kinds of infinite rewrite sequences.

### 1 Introduction

Term rewriting is an important method for equational reasoning. In term rewriting the axioms of the equational system under consideration are used in one direction only. Since in the presence of axioms like commutativity, a common situation in equational reasoning is non-terminating, the framework

## Theorem

AC termination is persistent for TRSs that do not contain both collapsing and duplicating rules

## Type Introduction for Equational Rewriting

Hitoshi Ohsaki and Aart Middeldorp

Institute of Information Sciences and Electronics  
University of Tsukuba  
Tsukuba 305, Japan  
{hitoshi,ami}@score.is.tsukuba.ac.jp

**Abstract.** Type introduction is a useful technique for simplifying the task of proving properties of rewrite systems by restricting the set of terms that have to be considered to the well-typed terms according to any many-sorted type discipline which is compatible with the rewrite system under consideration. A property of rewrite systems for which type introduction is correct is called persistent. Zantema showed that termination is a persistent property of non-collapsing rewrite systems and non-duplicating rewrite systems. We extend his result to the more complicated case of equational rewriting. As a simple application we prove the undecidability of AC-termination for terminating rewrite systems. We also present sufficient conditions for the persistence of acyclicity and non-loopingness, two properties which guarantee the absence of certain kinds of infinite rewrite sequences.

### 1 Introduction

Term rewriting is an important method for equational reasoning. In term rewriting the axioms of the equational system under consideration are used in one direction only. Since in the presence of axioms like commutativity, a common situation in equational reasoning is non-terminating the framework

## Theorem

AC termination is persistent for TRSs that do not contain both collapsing and duplicating rules



## Type Introduction for Equational Rewriting

Hitoshi Ohsaki and Aart Middeldorp

Institute of Information Sciences and Electronics  
University of Tsukuba  
Tsukuba 305, Japan  
{hitoshi,ami}@score.is.tsukuba.ac.jp

**Abstract.** Type introduction is a useful technique for simplifying the task of proving properties of rewrite systems by restricting the set of terms that have to be considered to the well-typed terms according to any many-sorted type discipline which is compatible with the rewrite system under consideration. A property of rewrite systems for which type introduction is correct is called persistent. Zantema showed that termination is a persistent property of non-collapsing rewrite systems and non-duplicating rewrite systems. We extend his result to the more complicated case of equational rewriting. As a simple application we prove the undecidability of AC-termination for terminating rewrite systems. We also present sufficient conditions for the persistence of acyclicity and non-loopingness, two properties which guarantee the absence of certain kinds of infinite rewrite sequences.

### 1 Introduction

Term rewriting is an important method for equational reasoning. In term rewriting the axioms of the equational system under consideration are used in one direction only. Since in the presence of axioms like commutativity, a common situation in equational reasoning is non-terminating the framework

# Outline

1. Hans Zantema
- 2. WST & IWC**
3. Hans Zantema
4. Workshops and Competitions
5. Hans Zantema

WST	
1993	St. Andrews
1995	La Bresse
1997	Ede
1999	Dagstuhl
2001	Utrecht
2003	Valencia
2004	Aachen
2006	Seattle
2007	Paris
2009	Leipzig
2010	Edinburgh
2012	Obergurgl

WST	
2013	Bertinoro
2014	Vienna
2015	
2016	Obergurgl
2017	
2018	Oxford
2019	
2020	
2021	Pittsburgh
2022	Haifa
2023	Obergurgl
2024	
2025	Leipzig



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig





- Termination of Term Rewriting by Semantic Labelling

## ► Termination of Term Rewriting by Semantic Labelling

### Termination of term rewriting by semantic labelling

H. Zantema

Utrecht University, Department of Computer Science,  
P.O. box 80.089, 3508 TB Utrecht, The Netherlands  
e-mail: hanzs@cs.ruu.nl

#### Abstract

A new kind of transformation of term rewriting systems (TRS) is proposed, depending on a choice for a model for the TRS. The labelled TRS is obtained from the original one by labelling operation symbols, possibly creating extra copies of some rules. This construction has the remarkable property that the labelled TRS is terminating if and only if the original TRS is terminating. Although the labelled version has more operation symbols and may have more rules (sometimes infinitely many), termination is often easier to prove for the labelled TRS than for the original one. This provides a new technique for proving termination, making classical techniques like path orders and polynomial interpretations applicable even for non-simplifying TRS's. The requirement of having a model can slightly be weakened, yielding a remarkably simple termination proof of the system SUBST of [11] describing explicit substitution in  $\lambda$ -calculus.

## 1 Introduction

- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness

## Transforming Termination by Self-Labelling

Aart Middeldorp,<sup>1</sup> Hitoshi Ohsaki,<sup>1</sup> Hans Zantema<sup>2</sup>

<sup>1</sup> Institute of Information Sciences and Electronics  
University of Tsukuba, Tsukuba 305, Japan

<sup>2</sup> Department of Computer Science, Utrecht University  
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

**Abstract.** We introduce a new technique for proving termination of term rewriting systems. The technique, a specialization of Zantema's semantic labelling technique, is especially useful for establishing the correctness of transformation methods that attempt to prove termination by transforming term rewriting systems into systems whose termination is easier to prove. We apply the technique to modularity, distribution elimination, and currying, resulting in new results, shorter correctness proofs, and a positive solution to an open problem.

### 1 Introduction

Termination is an undecidable property of term rewriting systems. In the literature (Dershowitz [4] contains an early survey of termination techniques) several methods for proving termination are described that are quite successful in practice. We can distinguish roughly two kinds of termination methods:

1. basic methods like recursive path order and polynomial interpretations that apply directly to a given term rewriting system, and
2. methods that attempt to prove termination by transforming a given term rewriting system into a term rewriting system whose termination is easier



- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness
- ▶ termination modulo

## Equational Termination by Semantic Labelling

Hitoshi Ohsaki<sup>1</sup>, Aart Middeldorp<sup>2</sup>, and Jürgen Giesl<sup>3</sup>

<sup>1</sup> Computer Science Division, Electrotechnical Laboratory  
Tsukuba 305-8568, Japan  
ohsaki@etl.go.jp

<sup>2</sup> Institute of Information Sciences and Electronics  
University of Tsukuba, Tsukuba 305-8573, Japan  
ami@is.tsukuba.ac.jp

<sup>3</sup> Computer Science Department  
University of New Mexico, Albuquerque, NM 87131, USA  
giesl@cs.unm.edu

**Abstract.** Semantic labelling is a powerful tool for proving termination of term rewrite systems. The usefulness of the extension to equational term rewriting described in Zantema [24] is however rather limited. In this paper we introduce a stronger version of *equational semantical labelling*, parameterized by three choices: (1) the order on the underlying algebra (partial order vs. quasi-order), (2) the relation between the algebra and the rewrite system (model vs. quasi-model), and (3) the labelling of the function symbols appearing in the equations (forbidden vs. allowed). We present soundness and completeness results for the various instantiations and analyze the relationships between them. Applications of our equational semantic labelling technique include a short proof of the main result of Ferreira *et al.* [7]—the correctness of a version of dummy elimination for AC rewriting which completely removes the AC-axioms—

- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness
- ▶ termination modulo
- ▶ predictive labeling

## Predictive Labeling

Nao Hirokawa and Aart Middeldorp

Institute of Computer Science  
University of Innsbruck  
6020 Innsbruck, Austria

{nao.hirokawa, aart.middeldorp}@uibk.ac.at

**Abstract.** Semantic labeling is a transformation technique for proving the termination of rewrite systems. The semantic part is given by a quasi-model of the rewrite rules. In this paper we present a variant of semantic labeling in which the quasi-model condition is only demanded for the usable rules induced by the labeling. Our variant is less powerful in theory but maybe more useful in practice.

- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness
- ▶ termination modulo
- ▶ predictive labeling
- ▶ automation

## Predictive Labeling with Dependency Pairs Using SAT

Adam Koprowski<sup>1</sup> and Aart Middeldorp<sup>2,\*</sup>

<sup>1</sup> Department of Computer Science  
Eindhoven University of Technology

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

<sup>2</sup> Institute of Computer Science  
University of Innsbruck  
6020 Innsbruck, Austria

**Abstract.** This paper combines predictive labeling with dependency pairs and reports on its implementation. Our starting point is the method of proving termination of rewrite systems using semantic labeling with infinite models in combination with lexicographic path orders. We replace semantic labeling with predictive labeling to weaken the quasi-model constraints and we combine it with dependency pairs (usable rules and argument filtering) to increase the power of the method. Encoding the resulting search problem as a propositional satisfiability problem and calling a state-of-the-art SAT solver yields a powerful technique for proving termination automatically.

- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness
- ▶ termination modulo
- ▶ predictive labeling
- ▶ automation

## Root-Labeling\*

Christian Sternagel and Aart Middeldorp

Institute of Computer Science  
University of Innsbruck  
Austria

**Abstract.** In 2006 *Jambox*, a termination prover developed by Endrullis, surprised the termination community by winning the string rewriting division and almost beating *AProVE* in the term rewriting division of the international termination competition. The success of *Jambox* for strings is partly due to a very special case of semantic labeling. In this paper we integrate this technique, which we call root-labeling, into the dependency pair framework. The result is a simple processor with help of which  $T_T T_2$  surprised the termination community in 2007 by producing the first automatically generated termination proof of a string rewrite system with non-primitive recursive complexity (Touzet, 1998). Unlike many other recent termination methods, the root-labeling processor is trivial to automate and completely unsuitable for producing human readable proofs.

- ▶ Termination of Term Rewriting by Semantic Labelling
- ▶ completeness
- ▶ termination modulo
- ▶ predictive labeling
- ▶ automation
- ▶ many-sorted AC version used in termination proof of faithful encoding of Battle of Hercules and Hydra

## HYDRA BATTLES AND AC TERMINATION

NAO HIROKAWA  AND AART MIDDELDORP 

<sup>a</sup>School of Information Science, JAIST, Japan  
*e-mail address:* hirokawa@jaist.ac.jp

<sup>b</sup>Department of Computer Science, University of Innsbruck, Austria  
*e-mail address:* aart.middeldorp@uibk.ac.at

**ABSTRACT.** We present a new encoding of the Battle of Hercules and Hydra as a rewrite system with AC symbols. Unlike earlier term rewriting encodings, it faithfully models any strategy of Hercules to beat Hydra. To prove the termination of our encoding, we employ type introduction in connection with many-sorted semantic labeling for AC rewriting and AC-MPO, a new AC compatible reduction order that can be seen as a much weakened version of AC-RPO.

## 1. INTRODUCTION

The mythological monster Hydra is a dragon-like creature with multiple heads. Whenever Hercules in his fight chops off a head, more and more new heads can grow instead, since the beast gets increasingly angry. Here we model a Hydra as an unordered tree. If Hercules cuts off a leaf corresponding to a head, the tree is modified in the following way: If the cut-off node  $h$  has a grandparent  $n$ , then the branch from  $n$  to the parent of  $h$  gets multiplied, where the number of copies corresponds to the number of decapitations so far. Hydra dies if there are no heads left, in that case Hercules wins. The following sequence shows an example fight:



# WST 1993

▶ Enno Ohlebusch

On the Modularity of Termination of  
Term Rewriting Systems

► Enno Ohlebusch

## On the Modularity of Termination of Term Rewriting Systems

Theoretical Computer Science 136 (1994) 333–360  
Elsevier

## On the modularity of termination of term rewriting systems

Enno Ohlebusch

*Universität Bielefeld, Technische Fakultät, Postfach 100131, 33501 Bielefeld, Germany*

Communicated by M. Takahashi

Received April 1993

Revised February 1994

### Abstract

Ohlebusch, E., On the modularity of termination of term rewriting systems, Theoretical Computer Science 136 (1994) 333–360.

It is well-known that termination is not a modular property of term rewriting systems, i.e., it is not preserved under disjoint union. The objective of this paper is to provide a “uniform framework” for sufficient conditions which ensure the modularity of termination. We will prove the following result. Whenever the disjoint union of two terminating term rewriting systems is nonterminating, then one of the systems is not  $\mathcal{C}_A$ -terminating (i.e., it loses its termination property when extended with the rules  $\text{Cons}(x, y) \rightarrow x$  and  $\text{Cons}(x, y) \rightarrow y$ ) and the other is collapsing. This result has already been achieved by Gramlich [7] for *finitely branching* term rewriting systems. A more sophisticated approach to the problem of modularity of termination is given in [8].

## ► Enno Ohlebusch

### On the Modularity of Termination of Term Rewriting Systems



## On the modularity of termination of term rewriting systems

Enno Ohlebusch

*Universität Bielefeld, Technische Fakultät, Postfach 100131, 33501 Bielefeld, Germany*

Communicated by M. Takahashi

Received April 1993

Revised February 1994

### Abstract

Ohlebusch, E., On the modularity of termination of term rewriting systems, Theoretical Computer Science 136 (1994) 333–360.

It is well-known that termination is not a modular property of term rewriting systems, i.e., it is not preserved under disjoint union. The objective of this paper is to provide a “uniform framework” for sufficient conditions which ensure the modularity of termination. We will prove the following result. Whenever the disjoint union of two terminating term rewriting systems is nonterminating, then one of the systems is not  $\mathcal{C}_A$ -terminating (i.e., it loses its termination property when extended with the rules  $\text{Cons}(x, y) \rightarrow x$  and  $\text{Cons}(x, y) \rightarrow y$ ) and the other is collapsing. This result has already been achieved by Gramlich [7] for *finitely branching* term rewriting systems. A more sophisticated approach to the problem of modularity of termination is given in [8].



# WST 1993

- ▶ Enno Ohlebusch

On the Modularity of Termination of  
Term Rewriting Systems

- ▶ Pierre Lescanne

Termination of Rewrite Systems by  
Elementary Interpretations

► Enno Ohlebusch

On the Modularity of Termination of  
Term Rewriting Systems

► Pierre Lescanne

Termination of Rewrite Systems by  
Elementary Interpretations

Formal Aspects of Computing (1995) 7: 77–90  
© 1995 BCS

**Formal Aspects  
of Computing**

## Termination of Rewrite Systems by Elementary Interpretations

Pierre Lescanne

Centre de Recherche en Informatique de Nancy (CNRS) and INRIA-Lorraine,  
Vandœuvre-lès-Nancy, France

**Keywords:** Rewrite systems; Termination; Well-foundedness; Total connection

**Abstract.** We focus on termination proofs of rewrite systems, especially of rewrite systems containing associative and commutative operators. We prove their termination by elementary interpretations, more specifically, by functions defined by addition, multiplication and exponentiation. We discuss a method based on polynomial interpretations and propose an implementation of a mechanisation of the comparison of expressions built with polynomials and exponentials.

- ▶ Enno Ohlebusch

On the Modularity of Termination of  
Term Rewriting Systems

- ▶ Pierre Lescanne

Termination of Rewrite Systems by  
Elementary Interpretations

- ▶ automation



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)



Electronic Notes in Theoretical Computer Science 258 (2009) 41–61

Electronic Notes in  
Theoretical Computer  
Science

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

## Automatic Proofs of Termination With Elementary Interpretations

Salvador Lucas<sup>1,2</sup>

Departamento de Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Valencia, Spain

### Abstract

*Symbolic constraints* arising in proofs of termination of programs are often translated into *numeric constraints* before checking them for satisfiability. In this setting, polynomial interpretations are a simple and popular choice. In the nineties, Lescanne introduced the *elementary algebraic interpretations* as a suitable alternative to polynomial interpretations in proofs of termination of term rewriting. Here, not only addition and product but also exponential expressions are allowed. Lescanne investigated the use of elementary interpretations for witnessing satisfiability of a given set of symbolic constraints. He also motivated the usefulness of elementary interpretations in proofs of termination by means of several examples. Unfortunately, he did not consider the *automatic generation* of such interpretations for a given termination problem. This is an important drawback for using these interpretations in practice. In this paper we show how to solve this problem by using a combination of rewriting, CLP, and CSP techniques for handling the elementary constraints which are obtained when giving the symbols *parametric elementary interpretations*.

**Keywords:** Constraint solving, elementary interpretations, program analysis, termination.

- ▶ Enno Ohlebusch  
On the Modularity of Termination of  
Term Rewriting Systems
- ▶ Pierre Lescanne  
Termination of Rewrite Systems by  
Elementary Interpretations
- ▶ automation
- ▶ Harald Zankl (WST 2014)  
Automating Elementary Interpretations



ELSEVIER

Contents lists available at [ScienceDirect](http://ScienceDirect.com)

Journal of Symbolic Computation

[www.elsevier.com/locate/jsc](http://www.elsevier.com/locate/jsc)



## Beyond polynomials and Peano arithmetic—automation of elementary and ordinal interpretations



Harald Zankl, Sarah Winkler, Aart Middeldorp

*Institute of Computer Science, University of Innsbruck, 6020 Innsbruck, Austria*

### ARTICLE INFO

#### Article history:

Received 31 January 2014

Accepted 25 June 2014

Available online 29 September 2014

#### Keywords:

Term rewriting

Termination

Automation

Ordinals

### ABSTRACT

Kirby and Paris (1982) proved in a celebrated paper that a theorem of Goodstein (1944) cannot be established in Peano arithmetic. We present an encoding of Goodstein's theorem as a termination problem of a finite rewrite system. Using a novel implementation of algebras based on ordinal interpretations, we are able to automatically prove termination of this system, resulting in the first automatic termination proof for a system whose derivational complexity is not multiple recursive. Our method can also cope with the encoding by Touzet (1998) of the battle of Hercules and Hydra as well as a (corrected) encoding by Beklemishev (2006) of the Worm battle, two further systems which have been out of reach for automatic tools, until now. Based on our ideas of implementing ordinal algebras we also present a new approach for the automation of elementary interpretations for termination

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## ► Total Termination of Term Rewriting is Undecidable

*J. Symbolic Computation* (1995) 20, 43–60

### Total Termination of Term Rewriting is Undecidable

HANS ZANTEMA

*Utrecht University, Department of Computer Science,  
P.O. box 80.089, 3508 TB Utrecht, The Netherlands <sup>†</sup>*

*(Received 7 February 1995)*

---

Usually termination of term rewriting systems (TRS's) is proved by means of a monotonic well-founded order. If this order is total on ground terms, the TRS is called totally terminating. In this paper we prove that total termination is an undecidable property of finite term rewriting systems. The proof is given by means of Post's Correspondence Problem.

---

- Total Termination of Term Rewriting is Undecidable



## Omega-Termination is Undecidable for Totally Terminating Term Rewriting Systems

ALFONS GESER<sup>†</sup>

*Wilhelm-Schickard-Institut für Informatik, Universität Tübingen,  
Sand 13, D-72076 Tübingen, Germany*

*(Received 11 April 1996)*

---

We give a complete proof of the fact that the following problem is undecidable:

**Given:** A term rewriting system, where the termination of its rewrite relation is provable by a total reduction order on ground terms,

**Wanted:** Does there exist a strictly monotonic interpretation in the positive integers that proves termination?

© 1997 Academic Press Limited

---

### 1. Introduction

Termination of a term rewriting system (TRS), i.e. the non-existence of an infinite rewrite reduction,  $t_1 \rightarrow_R t_2 \rightarrow_R \dots$ , is one of the key notions in term rewriting. It is the basis of a number of decision algorithms for properties undecidable in the general case. In this

## ► Total Termination of Term Rewriting is Undecidable

Information and Computation **178**, 101–131 (2002)  
doi:10.1006/inco.2002.3120

### Relative Undecidability in Term Rewriting

#### I. The Termination Hierarchy

Alfons Geser

*ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681*

Aart Middeldorp

*Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba 305-8573, Japan*

Enno Ohlebusch

*Faculty of Technology, University of Bielefeld, P.O. Box 10 01 31, 33501 Bielefeld, Germany*

and

Hans Zantema

*Department of Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Received February 10, 1999; revised March 9, 2000

For a hierarchy of properties of term rewriting systems related to termination we prove *relative undecidability*: For implications  $X \Rightarrow Y$  in the hierarchy the property  $X$  is undecidable for term rewriting systems satisfying  $Y$ . For most implications we obtain this result for term rewriting systems consisting of a single rewrite rule. © 2002 Elsevier Science (USA)

**Key Words:** term rewriting; termination; undecidability.



# WST 1995

- ▶ Total Termination of Term Rewriting is Undecidable
- ▶ w/ Thomas Arts:  
A New Technique to Prove Termination of Constructor Systems

- ▶ Total Termination of Term Rewriting is Undecidable

- ▶ w/ Thomas Arts:

## A New Technique to Prove Termination of Constructor Systems

### Termination of Constructor Systems\*

Thomas Arts<sup>1</sup> and Jürgen Giesl<sup>2</sup>

<sup>1</sup> Dept. of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands, E-mail: [thomas@cs.ruu.nl](mailto:thomas@cs.ruu.nl)

<sup>2</sup> FB Informatik, TH Darmstadt, Alexanderstr. 10, 64283 Darmstadt, Germany, E-mail: [giesl@inferenzsysteme.informatik.th-darmstadt.de](mailto:giesl@inferenzsysteme.informatik.th-darmstadt.de)

**Abstract.** We present a method to prove termination of constructor systems automatically. Our approach takes advantage of the special form of these rewrite systems because for constructor systems instead of left- and right-hand sides of rules it is sufficient to compare so-called *dependency pairs* [Art96]. Unfortunately, standard techniques for the generation of well-founded orderings cannot be directly used for the automation of the dependency pair approach. To solve this problem we have developed a transformation technique which enables the application of known synthesis methods for well-founded orderings to prove that dependency pairs are decreasing. In this way termination of many (also non-simply terminating) constructor systems can be proved fully automatically.

- ▶ Total Termination of Term Rewriting is Undecidable

- ▶ w/ Thomas Arts:

## A New Technique to Prove Termination of Constructor Systems



## Termination of Constructor Systems\*

Thomas Arts<sup>1</sup> and Jürgen Giesl<sup>2</sup>

<sup>1</sup> Dept. of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands, E-mail: [thomas@cs.ruu.nl](mailto:thomas@cs.ruu.nl)

<sup>2</sup> FB Informatik, TH Darmstadt, Alexanderstr. 10, 64283 Darmstadt, Germany, E-mail: [giesl@inferenzsysteme.informatik.th-darmstadt.de](mailto:giesl@inferenzsysteme.informatik.th-darmstadt.de)

**Abstract.** We present a method to prove termination of constructor systems automatically. Our approach takes advantage of the special form of these rewrite systems because for constructor systems instead of left- and right-hand sides of rules it is sufficient to compare so-called *dependency pairs* [Art96]. Unfortunately, standard techniques for the generation of well-founded orderings cannot be directly used for the automation of the dependency pair approach. To solve this problem we have developed a transformation technique which enables the application of known synthesis methods for well-founded orderings to prove that dependency pairs are decreasing. In this way termination of many (also non-simply terminating) constructor systems can be proved fully automatically.

- ▶ Total Termination of Term Rewriting is Undecidable
- ▶ w/ Thomas Arts:  
A New Technique to Prove Termination of Constructor Systems
- ▶ w/ Maria Ferreira:  
Dummy Elimination in Term Rewriting

## Dummy Elimination: Making Termination Easier

M. C. F. Ferreira\* and H. Zantema

tel: +31-30-532249, fax: +31-30-513791, e-mail: {maria, hansz}@cs.ruu.nl

Utrecht University, Department of Computer Science  
P.O. box 80.089, 3508 TB Utrecht, The Netherlands

**Abstract.** We investigate a technique whose goal is to simplify the task of proving termination of term rewriting systems. The technique consists of a transformation which eliminates function symbols considered “useless” and simplifies the rewrite rules. We show that the transformation is sound, i. e., termination of the original system can be inferred from termination of the transformed one. For proving this result we use a new notion of lifting of orders that is a generalization of the multiset construction.

- ▶ Total Termination of Term Rewriting is Undecidable
- ▶ w/ Thomas Arts:  
A New Technique to Prove Termination of Constructor Systems
- ▶ w/ Maria Ferreira:  
Dummy Elimination in Term Rewriting
- ▶ w/ Aart Middeldorp:  
Simple Termination of Rewrite Systems



ELSEVIER

Theoretical Computer Science 175 (1997) 127–158

Theoretical  
Computer Science

## Simple termination of rewrite systems<sup>1</sup>

Aart Middeldorp<sup>a,\*,2</sup>, Hans Zantema<sup>b</sup><sup>a</sup> Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba 305, Japan<sup>b</sup> Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

### Abstract

In this paper we investigate the concept of simple termination. A term rewriting system is called simply terminating if its termination can be proved by means of a simplification order. The basic ingredient of a simplification order is the subterm property, but in the literature two different definitions are given: one based on (strict) partial orders and another one based on preorders (or quasi-orders). We argue that there is no reason to choose the second one as the first one has certain advantages.

Simplification orders are known to be well-founded orders on terms over a finite signature. This important result no longer holds if we consider infinite signatures. Nevertheless, well-known simplification orders like the recursive path order are also well-founded on terms over infinite signatures, provided the underlying precedence is well-founded. We propose a new definition of simplification order, which coincides with the old one (based on partial orders) in case of finite signatures, but which is also well-founded over infinite signatures and covers orders like the recursive path order. We investigate the properties of the ensuing class of simply terminating systems.

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- Transformational Techniques for Proving Termination of Term Rewriting

## ► Transformational Techniques for Proving Termination of Term Rewriting

## ► Alfons Geser

### On Normalizing, Non-Terminating One-Rule String Rewriting Systems



ELSEVIER

Theoretical Computer Science 243 (2000) 489–498

TCS 2000

Theoretical  
Computer Science

[www.elsevier.com/locate/tcs](http://www.elsevier.com/locate/tcs)

Note

### On normalizing, non-terminating one-rule string rewriting systems

Alfons Geser<sup>1</sup>

*Alte Straße 42, D-94034 Passau, Germany*

Received November 1996; revised December 1999; accepted February 2000

#### Abstract

We prove that the one-rule string rewriting system  $1010 \rightarrow 010110$  is normalizing, i.e. admits for every string a reduction to normal form, but non-terminating, i.e. there are infinite reductions as well. Moreover, we prove that this is the smallest such system. Whereas  $1010 \rightarrow 010110$  is rightmost terminating, i.e. no infinite rightmost reductions exist, the normalizing system  $1^201^3 \rightarrow 01^60$  is neither leftmost terminating nor rightmost terminating. In the discourse, we introduce a few methods to prove or disprove normalization for non-terminating string rewriting systems. © 2000 Elsevier Science B.V. All rights reserved.

**Keywords:** String rewriting system; Semi-Thue system; Single-rule; One-rule; Rightmost terminating; Innermost terminating; Terminating; Normalizing



## ► Transformational Techniques for Proving Termination of Term Rewriting

### ► Alfons Geser

On Normalizing, Non-Terminating One-Rule String Rewriting Systems

### ► Johannes Waldmann

Deciding Termination in  $CL(S)$

## Normalization of S-Terms is Decidable

Johannes Waldmann

Institut für Informatik, Friedrich-Schiller-Universität, D-07740 Jena, Germany  
joe@informatik.uni-jena.de, <http://www5.informatik.uni-jena.de/~joe/>

**Abstract.** The combinator **S** has the reduction rule  $S\ x\ y\ z \rightarrow x\ z\ (y\ z)$ . We investigate properties of ground terms built from **S** alone. We show that it is decidable whether such an **S**-term has a normal form. The decision procedure makes use of rational tree languages. We also exemplify and summarize other properties of **S**-terms and hint at open questions.

### 1 Introduction

It is well known that the combinators **S** and **K** with their reduction rules  $S\ x\ y\ z \rightarrow x\ z\ (y\ z)$  and  $K\ x\ y \rightarrow x$  form a complete basis for combinatory logic. Therefore, most of the interesting properties of an  $(S, K)$ -term are necessarily undecidable.

Now  $CL(K)$  is strongly normalizing. One is lead to assume that the difficulty of  $CL(S, K)$  comes from **S**. That's the reason for studying the system  $CL(S)$ . Somewhat surprisingly it turns out that normalization is decidable for **S**-terms. (Since  $CL(S)$  is orthogonal and non-erasing, an **S**-term is normalizing (has a normal form) iff it is terminating (has no infinite reduction)).

We will start by giving some examples of infinite reductions in  $CL(S)$ .

## ► Transformational Techniques for Proving Termination of Term Rewriting

### ► Alfons Geser

On Normalizing, Non-Terminating One-Rule String Rewriting Systems

### ► Johannes Waldmann

Deciding Termination in  $CL(S)$

## Normalization of S-Terms is Decidable

Johannes Waldmann

Institut für Informatik, Friedrich-Schiller-Universität, D-07740 Jena, Germany  
joe@informatik.uni-jena.de, <http://www5.informatik.uni-jena.de/~joe/>

**Abstract.** The combinator **S** has the reduction rule  $S\ x\ y\ z \rightarrow x\ z\ (y\ z)$ . We investigate properties of ground terms built from **S** alone. We show that it is decidable whether such an **S**-term has a normal form. The decision procedure makes use of rational tree languages. We also exemplify and summarize other properties of **S**-terms and hint at open questions.

### 1 Introduction

It is well known that the combinators **S** and **K** with their reduction rules  $S\ x\ y\ z \rightarrow x\ z\ (y\ z)$  and  $K\ x\ y \rightarrow x$  form a complete basis for combinatory logic. Therefore, most of the interesting properties of an  $(S, K)$ -term are necessarily undecidable.

Now **CL(S)** is orthogonal and non-erasing, an **S**-term is normalizing (has a normal form) iff it is terminating (has no infinite reduction)).

We will start by giving two examples of infinite reductions in **CL(S)**.

best student paper award

- ▶ Transformational Techniques for Proving Termination of Term Rewriting
- ▶ Alfons Geser  
On Normalizing, Non-Terminating One-Rule String Rewriting Systems
- ▶ Johannes Waldmann  
Deciding Termination in  $CL(S)$

## The Combinator S

Johannes Waldmann

*Institut für Informatik, Universität Leipzig, Augustusplatz 10–11, D-04109 Leipzig, Germany*

E-mail: [joe@informatik.uni-leipzig.de](mailto:joe@informatik.uni-leipzig.de)

URL: <http://www.informatik.uni-leipzig.de/~joe/>

---

The combinator  $S$  has the reduction rule  $Sx y z \rightarrow xz (yz)$ . We investigate properties of ground terms built from  $S$  alone. The first part of the paper shows that this term rewriting system admits no ground loops. This extends the known result of the absence of cycles. In the second part, we give a procedure that decides whether an  $S$ -term has a normal form. This algorithm makes use of rational tree languages. Finally we show that the set of normalizing  $S$ -terms is in itself a rational tree language, by explicitly giving its grammar. In all, this paper shows the surprisingly rich structures that are implied by a seemingly small rewrite rule. © 2000 Academic Press

---

- ▶ Transformational Techniques for Proving Termination of Term Rewriting
- ▶ Alfons Geser  
On Normalizing, Non-Terminating One-Rule String Rewriting Systems
- ▶ Johannes Waldmann  
Deciding Termination in  $CL(S)$
- ▶ Pierre Lescanne  
Strong Normalisation in 2nd Order Lambda Calculus with Explicit Substitutions

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## ► The Termination Hierarchy for Term Rewriting

*J. Symbolic Computation* (1994) 17, 23–50

### Termination of term rewriting: interpretation and type elimination

H. Zantema

*Utrecht University, Department of Computer Science*

*P.O. box 80.089, 3508 TB Utrecht, The Netherlands*

*e-mail: hansz@cs.ruu.nl*

*(Received 25 March 1993)*

We investigate proving termination of term rewriting systems by interpretation of terms in a well-founded monotone algebra. The well-known polynomial interpretations can be considered as a particular case in this framework. A classification of types of termination, including simple termination, is proposed based on properties in the semantic level. A transformation on term rewriting systems eliminating distributive rules is introduced. Using this distribution elimination a new termination proof of the system SUBST of Hardin and Laville (1986) is given. This system describes explicit substitution in  $\lambda$ -calculus.

Another tool for proving termination is based on introduction and removal of type restrictions. A property of many-sorted term rewriting systems is called persistent if it is not affected by removing the corresponding typing restriction. Persistence turns out to be a generalization of distributive sum modularity but is more powerful for both

► The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

*J. Symbolic Computation* (1994) 17, 23–50

## Termination of term rewriting: interpretation and type elimination

H. Zantema

*Utrecht University, Department of Computer Science*

*P.O. box 80.089, 3508 TB Utrecht, The Netherlands*

*e-mail: hansz@cs.ruu.nl*

*(Received 25 March 1993)*

We investigate proving termination of term rewriting systems by interpretation of terms in a well-founded monotone algebra. The well-known polynomial interpretations can be considered as a particular case in this framework. A classification of types of termination, including simple termination, is proposed based on properties in the semantic level. A transformation on term rewriting systems eliminating distributive rules is introduced. Using this distribution elimination a new termination proof of the system SUBST of Hardin and Laville (1986) is given. This system describes explicit substitution in  $\lambda$ -calculus.

Another tool for proving termination is based on introduction and removal of type restrictions. A property of many-sorted term rewriting systems is called persistent if it is not affected by removing the corresponding typing restriction. Persistence turns out to be a generalization of distributive sum modularity but is more powerful for both

## ► The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

AAECC 12, 3–19 (2001)



## The Termination Hierarchy for Term Rewriting

**H. Zantema**

Department of Computing Science, Eindhoven University of Technology, P.O. Box 513,  
5600 MB Eindhoven, The Netherlands (e-mail: h.zantema@tue.nl)

Received: June 15, 1999; revised version: August 7, 2000

**Abstract.** A number of properties of term rewriting systems related to termination are discussed. It is examined how these properties are affected by modifications in the definitions like weakening the requirement of strict monotonicity and adding embedding rules. All counterexamples to prove non-equivalence of properties are string rewriting systems, in most cases even single string rewrite rules.

**Keywords:** Term rewriting, Termination, String rewriting, Monotone algebras.

### 1 Introduction



- ▶ The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

- ▶ Dieter Hofbauer

On Termination and Derivation Lengths for  
Ground Rewrite Systems

## ► The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

## ► Dieter Hofbauer

On Termination and Derivation Lengths for Ground Rewrite Systems

### Termination Proofs for Ground Rewrite Systems – Interpretations and Derivational Complexity

**Dieter Hofbauer**

Universität GH Kassel, Fachbereich 17 Mathematik/Informatik, 34109 Kassel, Germany  
(e-mail: dieter@theory.informatik.uni-kassel.de)

Received: December 8, 1999

**Abstract.** It is shown that finite terminating ground term rewrite systems have linearly bounded derivational complexity. This is proven via some suitable interpretation into the natural numbers. Terminating ground systems are not necessarily totally terminating, i.e., a strictly monotone interpretation into a well-order might not exist. We show, however, that those systems are *almost  $\omega$ -terminating* in the sense that such an interpretation into the sum of a finite partial order and the natural numbers always effectively exists. Finally, we show that for ground systems total termination is equivalent to  $\omega$ -termination, and that this is a decidable property.

- ▶ The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

- ▶ Dieter Hofbauer  
On Termination and Derivation Lengths for  
Ground Rewrite Systems
- ▶ Jürgen Giesl, Vincent van Oostrom, Fer-Jan de Vries  
Strong Convergence of Term Rewriting using  
Strong Dependency Pairs

# WST 1999

- ▶ The Termination Hierarchy for Term Rewriting

$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN$

- ▶ Dieter Hofbauer

On Termination and Derivation Lengths for  
Ground Rewrite Systems

- ▶ Jürgen Giesl, Vincent van Oostrom, Fer-Jan de Vries  
Strong Convergence of Term Rewriting using  
Strong Dependency Pairs



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## Decidability of Termination of Certain One-Rule String Rewriting Systems

### Termination of String Rewriting Rules That Have One Pair of Overlaps\*

Alfons Geser\*\*

National Institute for Aerospace, Hampton VA, USA\*\*\*

**Abstract.** This paper presents a partial solution to the long standing open problem whether termination of one-rule string rewriting is decidable. Overlaps between the two sides of the rule play a central role in existing termination criteria. We characterize termination of all one-rule string rewriting systems that have one such overlap at either end. This both completes a result of Kurth and generalizes a result of Shikishima-Tsuji et al.

**Key Words and Phrases:** semi-Thue system, string rewriting, one-rule, single-rule, termination, uniform termination, overlap

#### 1 Introduction and Related Work

Whether termination of one-rule string rewriting systems (SRSs) is decidable or not, is a long standing open problem [2,3,4,6,10,11,12,13,14,15,17]. A systematic approach was started by Kurth [7]. Kurth introduced a number of termination criteria to decide termination for all  $\ell \rightarrow r$  where  $|r| \leq 6$ .<sup>1</sup>

Most of Kurth's criteria (5 out of 8), and indeed most of the criteria introduced since, have been based on the set of overlaps of the left hand side (from

## ► Alfons Geser

Decidability of Termination of Certain  
One-Rule String Rewriting Systems

## ► Zantema's Problem

0011  $\rightarrow$  111000

## Termination of String Rewriting Rules That Have One Pair of Overlaps\*

Alfons Geser\*\*

National Institute for Aerospace, Hampton VA, USA\*\*\*

**Abstract.** This paper presents a partial solution to the long standing open problem whether termination of one-rule string rewriting is decidable. Overlaps between the two sides of the rule play a central role in existing termination criteria. We characterize termination of all one-rule string rewriting systems that have one such overlap at either end. This both completes a result of Kurth and generalizes a result of Shikishima-Tsuji et al.

**Key Words and Phrases:** semi-Thue system, string rewriting, one-rule, single-rule, termination, uniform termination, overlap

## 1 Introduction and Related Work

Whether termination of one-rule string rewriting systems (SRSs) is decidable or not, is a long standing open problem [2,3,4,6,10,11,12,13,14,15,17]. A systematic approach was started by Kurth [7]. Kurth introduced a number of termination criteria to decide termination for all  $\ell \rightarrow r$  where  $|r| \leq 6$ .<sup>1</sup>

Most of Kurth's criteria (5 out of 8), and indeed most of the criteria introduced since, have been based on the set of overlaps of the left hand side (from

## ► Alfons Geser

### Decidability of Termination of Certain One-Rule String Rewriting Systems

## ► Zantema's Problem

0011  $\rightarrow$  111000

AAECC 11, 1–25 (2000)



### A Complete Characterization of Termination of $0^p 1^q \rightarrow 1^r 0^s$

Hans Zantema<sup>1</sup>, Alfons Geser<sup>2,\*</sup>

<sup>1</sup>Universiteit Utrecht, P.O. Box 80089, 3508 Utrecht, The Netherlands  
(e-mail: hansz@cs.ruu.nl)

<sup>2</sup>Alte Strasse 42, 94034 Passau, Germany (e-mail: geser@fmi.uni-passau.de)

Received: July 31, 1997; revised version: January 30, 2000

**Abstract.** We characterize termination of one-rule string rewriting systems of the form  $0^p 1^q \rightarrow 1^r 0^s$  for every choice of positive integers  $p, q, r$ , and  $s$ . In doing so we introduce a termination proof method that applies to some hard examples. For the simply terminating cases, i.e. string rewriting systems that can be ordered by a division order, we give the precise complexity of derivation lengths.

**Keywords:** String rewriting, Term rewriting, Termination, Simple termination, Transformation order, Dummy elimination, Derivation length.



► Alfons Geser

Decidability of Termination of Certain  
One-Rule String Rewriting Systems

► Zantema's Problem

0011  $\rightarrow$  111000

## On the Termination Problem for One-Rule Semi-Thue System

Géraud Sénizergues

LaBRI

Université de Bordeaux I

351, Cours de la Libération 33405 Talence, France \*\*

**Abstract.** We solve the u-termination and the termination problems for the one-rule semi-Thue systems  $S$  of the form  $0^p 1^q \rightarrow v$ ,  $(p, q \in \mathbb{N} - \{0\}, v \in \{0, 1\}^*)$ . We obtain a structure theorem about a monoid that we call the *termination-monoid* of  $S$ . As a consequence, for every fixed system  $S$  of the above form, the termination-problem has a linear time-complexity.

**Keywords:** semi-Thue systems; termination; finite automata; rational monoid; automatic structure;

### 1 Introduction

The semi-Thue systems are among the simplest type of rewriting systems that occur in the literature: they can be seen as the special case of term rewriting systems where function symbols are of arity 1 only, which implies also that they are linear (with only one variable). Despite this apparent simplicity it has been shown recently that the *termination* and *u-termination* problems (see §2.4 for precise definitions) are both *undecidable* for 3 rules semi-Thue systems ([MS96]). In the case of semi-Thue systems with only one rule, these two algorithmic problems are still open. Partial results are obtained in [Kur90, Ges91, DH93, McN94, ZG95].

We show here that for the one rule semi-Thue systems of the form  $0^p 1^q \rightarrow v$ ,  $(p, q \in \mathbb{N} - \{0\}, v \in \{0, 1\}^*)$ , these two problems are decidable, thus generalizing the results of [ZG95].

The general idea of our solution is that the termination (resp. u-termination) problem for a system  $u \rightarrow v$  can be reduced to the study of either the set of irreducible words  $L_+(S) = \{\beta \in Irr(S) \mid \rho(\beta u) = \infty\}$  or its analogue on the right  $R_+(S) = \{\gamma \in Irr(S) \mid \rho(u\gamma) = \infty\}$  (provided  $u$  has no self-overlap). We show that for semi-Thue systems of the above form, it is always true that  $L_+(S) \neq \emptyset$  (resp.  $R_+(S) \neq \emptyset$ ) if and only if  $u$  is not a power of a word.

## ▶ Alfons Geser

### Decidability of Termination of Certain One-Rule String Rewriting Systems

## ▶ Zantema's Problem

0011 → 111000



ELSEVIER

Theoretical Computer Science 262 (2001) 583–632

TCS 2001

Theoretical  
Computer Science

[www.elsevier.com/locate/tcs](http://www.elsevier.com/locate/tcs)

### Termination and derivational complexity of confluent one-rule string-rewriting systems

Yuji Kobayashi<sup>a,\*</sup>, Masashi Katsura<sup>b</sup>, Kayoko Shikishima-Tsuji<sup>c</sup>

<sup>a</sup>*Department of Information Science, Faculty of Science, Toho University, Funabashi 274, Japan*

<sup>b</sup>*Department of Mathematics, Kyoto Sangyo University, Kyoto 603, Japan*

<sup>c</sup>*Faculty of Liberal Arts, Tenri University, Tenri 632, Japan*

Received 15 May 1997; revised 15 May 2000; accepted 7 August 2000

Communicated by G. Rozenberg

#### Abstract

It is not known whether the termination problem is decidable for one-rule string-rewriting systems, though the confluence of such systems is decidable by Wrathall (in: Word Equations and Related Topics, Lecture Notes in Computer Science, vol. 572, 1992, pp. 237–246). In this paper we develop techniques to attack the termination and complexity problems of confluent one-rule string-rewriting systems. With given such a system we associate another rewriting system over another alphabet. The behaviour of the two systems is closely related and the termination problem for the new system is sometimes easier than for the original system. We apply our method to systems of the special type  $\{a^p b^q \rightarrow t\}$ , where  $t$  is an arbitrary word over  $\{a, b\}$ , and give a complete characterization for termination. We also give a complete analysis of the derivational complexity for the system  $\{a^p b^q \rightarrow b^q a^p\}$ . © 2001 Elsevier Science B.V. All rights reserved.

**Keywords:** Termination problem; String rewriting system; Derivational complexity

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## Proving Liveness in Ring Protocols by Termination

### Liveness in Rewriting

Jürgen Giesl<sup>1</sup> and Hans Zantema<sup>2</sup>

<sup>1</sup> LuFG Informatik II, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany,  
giesl@informatik.rwth-aachen.de

<sup>2</sup> Department of Computer Science, TU Eindhoven, P.O. Box 513,  
5600 MB Eindhoven, The Netherlands, h.zantema@tue.nl

**Abstract.** In this paper, we show how the problem of verifying liveness properties is related to termination of term rewrite systems (TRSs). We formalize liveness in the framework of rewriting and present a sound and complete transformation to transform particular liveness problems into TRSs. Then the transformed TRS terminates if and only if the original liveness property holds. This shows that liveness and termination are essentially equivalent. To apply our approach in practice, we introduce a simpler sound transformation which only satisfies the ‘only if’-part. By refining existing techniques for proving termination of TRSs we show how liveness properties can be verified automatically. As examples, we prove a liveness property of a waiting line protocol for a network of processes and a liveness property of a protocol on a ring of processes.

### 1 Introduction

Usually, *liveness* is roughly defined as: “*something will eventually happen*” [1] and it is often remarked that “*termination is a particular case of liveness*”. In this paper we present liveness in the general but precise setting of abstract reduction

- ▶ w/ Jürgen Giesl:

Proving Liveness in Ring Protocols by  
Termination

- ▶ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

Match-Bounded String Rewriting and  
Automated Termination Proofs

- ▶ w/ Jürgen Giesl:

## Proving Liveness in Ring Protocols by Termination

- ▶ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

## Match-Bounded String Rewriting and Automated Termination Proofs

AAECC 15, 149–171 (2004)

Digital Object Identifier (DOI) 10.1007/s00200-004-0162-8



### Match-Bounded String Rewriting Systems

Alfons Geser<sup>1,\*</sup>, Dieter Hofbauer<sup>2</sup>, Johannes Waldmann<sup>3</sup>

<sup>1</sup>National Institute of Aerospace, 144 Research Drive, Hampton, VA 23666, USA

(e-mail: geser@nianet.org)

<sup>2</sup>Mühlengasse 16, 34125 Kassel, Germany (e-mail: dieter@theory.informatik.uni-kassel.de)

<sup>3</sup>Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fb IMN, PF 30 11 66, 04251 Leipzig, Germany (e-mail: waldmann@imn.htwk-leipzig.de)

Received: October 13, 2003; revised version: June 7, 2004

Published online: October 14, 2004 – © Springer-Verlag 2004

**Abstract.** We introduce a new class of automated proof methods for the termination of rewriting systems on strings. The basis of all these methods is to show that rewriting preserves regular languages. To this end, letters are annotated with natural numbers, called *match heights*. If the minimal height of all positions in a redex is  $h$  then every position in the reduct will get height  $h + 1$ . In a *match-bounded* system, match heights are globally bounded. Using recent results on *deleting* systems, we prove that rewriting by a match-bounded system preserves regular languages. Hence it is decidable whether a given rewriting

- ▶ w/ Jürgen Giesl:

## Proving Liveness in Ring Protocols by Termination

- ▶ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

## Match-Bounded String Rewriting and Automated Termination Proofs

### Matchbox: A Tool for Match-Bounded String Rewriting

Johannes Waldmann

Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig  
Fachbereich IMN, Postfach 30 11 66, D-04251 Leipzig, Germany  
waldmann@imn.htwk-leipzig.de

**Abstract.** The program **Matchbox** implements the exact computation of the set of descendants of a regular language, and of the set of non-terminating strings, with respect to an (inverse) match-bounded string rewriting system. **Matchbox** can search for proof or disproof of a Boolean combination of match-height properties of a given rewrite system, and some of its transformed variants. This is applied in various ways to search for proofs of termination and non-termination. **Matchbox** is the first program that delivers automated proofs of termination for some difficult string rewriting systems.

#### 1 Introduction

The theory of match bounded string rewriting, recently developed in [4, 6–8], allows to apply methods from formal language theory to problems of string rewriting. The basic result is a decomposition theorem. It implies effective preservation of regular languages, and effective regularity of the set of non-terminating strings w.r.t. the inverse system. The program **Matchbox** contains implementations of all of the fundamental algorithms, and allows to apply them in various settings. Given a rewrite system, **Matchbox** searches for proofs and disproofs of a Boolean

- ▶ w/ Jürgen Giesl:  
Proving Liveness in Ring Protocols by Termination
- ▶ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann  
Match-Bounded String Rewriting and  
Automated Termination Proofs
- ▶ tool demonstration session leading to  
Termination Competition from 2004



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## ► Relative Termination in Term Rewriting

- ▶ Relative Termination in Term Rewriting

- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

Tree Automata that Certify Termination of  
Term Rewriting Systems

## ► Relative Termination in Term Rewriting

► w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

## Tree Automata that Certify Termination of Term Rewriting Systems



### On tree automata that certify termination of left-linear term rewriting systems

Alfons Geser <sup>a,\*</sup>, Dieter Hofbauer <sup>b</sup> Johannes Waldmann <sup>c</sup> Hans Zantema <sup>d</sup>

<sup>a</sup>*Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fachbereich EIT, Postfach 301166,  
D-04251 Leipzig, Germany*

<sup>b</sup>*Mühlengasse 16, D-34125 Kassel, Germany*

<sup>c</sup>*Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fachbereich IMN, Postfach 301166,  
D-04251 Leipzig, Germany*

<sup>d</sup>*Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, Postbus 513,  
5600 MB Eindhoven, The Netherlands*

Received 19 September 2005; revised 21 June 2006

Available online 2 February 2007

#### Abstract

We present a new method for automatically proving termination of left-linear term rewriting systems on a given regular language of terms. It is a generalization of the match bound method for string rewriting. To prove that a term rewriting system terminates we first construct an enriched system over a new signature that simulates the original derivations. The enriched system is an infinite system over an infinite signature, but it is locally terminating: every restriction of the enriched system to a finite signature is terminating. We then construct iteratively a finite tree automaton that accepts the enriched given regular language and is closed under rewriting modulo the enriched system. If this procedure stops, then the enriched system is compact: every enriched derivation involves only a finite signature. Therefore, the original system terminates. We present two methods to construct the automaton: one for left-linear systems and one for non-linear systems.

- ▶ Relative Termination in Term Rewriting

- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

Tree Automata that Certify Termination of  
Term Rewriting Systems

- ▶ Salvador Lucas

Polynomials over the Reals in Proofs of  
Termination

- ▶ Relative Termination in Term Rewriting
- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann
- Tree Automata that Certify Termination of  
Term Rewriting Systems
- ▶ Salvador Lucas
- Polynomials over the Reals in Proofs of  
Termination

AAECC (2006) 17: 49–73  
DOI 10.1007/s00200-005-0189-5

Salvador Lucas

## On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting

Received: 11 March 2005 / Revised: 1 June 2005 / Published online: 6 March 2006  
© Springer-Verlag 2006

**Abstract** In the seventies, Manna and Ness, Lankford, and Dershowitz pioneered the use of polynomial interpretations with integer and real coefficients in proofs of termination of rewriting. More than twenty five years after these works were published, however, the absence of true examples in the literature has given rise to some doubts about the possible benefits of using polynomials with real or rational coefficients. In this paper we prove that there are, in fact, rewriting systems that can be proved polynomially terminating by using polynomial interpretations with (algebraic) real coefficients; however, the proof cannot be achieved if polynomials only contain rational coefficients. This result is proved by using a new technique, the so-called “relative termination”.

- ▶ Relative Termination in Term Rewriting

- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann

Tree Automata that Certify Termination of  
Term Rewriting Systems

- ▶ Salvador Lucas

Polynomials over the Reals in Proofs of  
Termination

- ▶ three system descriptions  
(AProVE, TTT, MuTerm)

- ▶ **Relative Termination in Term Rewriting**
- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann  
**Tree Automata that Certify Termination of  
Term Rewriting Systems**
- ▶ Salvador Lucas  
Polynomials over the Reals in Proofs of  
Termination
- ▶ three system descriptions  
(AProVE, TTT, MuTerm)
- ▶ **Torpa** wins SRS category of TermComp  
2004 and 2005



- ▶ **Relative Termination in Term Rewriting**
- ▶ w/ Alfons Geser, Dieter Hofbauer  
Johannes Waldmann  
**Tree Automata that Certify Termination of Term Rewriting Systems**
- ▶ Salvador Lucas  
**Polynomials over the Reals in Proofs of Termination**
- ▶ three system descriptions  
(AProVE, TTT, MuTerm)
- ▶ **Torpa** wins SRS category of TermComp 2004 and 2005

## Termination of String Rewriting Proved Automatically

H. ZANTEMA

*Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands. e-mail: h.zantema@tue.nl*

**Abstract.** This paper describes how a combination of polynomial interpretations, recursive path order, RFC match-bounds, the dependency pair method, and semantic labelling can be used for automatically proving termination of an extensive class of string rewriting systems (SRSs). The tool implementing this combination of techniques is called TORPA: Termination of Rewriting Proved Automatically. All termination proofs generated by TORPA are easy to read and check; but for many of the SRSs involved, finding a termination proof would be a hard job for a human. This paper contains all underlying theory, describes how the search for a termination proof is implemented, and includes many examples.

**Key words:** string rewriting, termination, semantic labelling, relative termination, match-bounds.

### 1. Introduction

In the past few decades many techniques have been developed for proving termi-

*Het onderzoek in termherschrijven maakte ook zijn ontwikkelingen door. In de jaren negentig lag bij het ontwikkelen van technieken om bijvoorbeeld terminatie te bewijzen, de nadruk puur op de theorie: je beschreef een nieuwe techniek, gaf er enkele voorbeelden bij, en een artikel daarover kon je publiceren. Rond 2003 kwam de lat een stuk hoger te liggen: er werden tools ontwikkeld waarmee terminatiebewijzen automatisch gegeven konden worden, en een data base aan herschrijfsystemen waar je die tools op los kon laten. En jaarlijks was er een competitie, een wedstrijd waar gekeken werd hoe goed die tools scoorden op de voorbeelden uit die data base. Vanaf dat moment kwam je met een nieuwe techniek alleen maar weg als je die in een tool implementeerde, en dat tool daarmee goed scoorde in die competitie. Beperkt tot de categorie van stringherschrijfsystemen heb ik hier ook aan mee gedaan, en enkele keren gewonnen. Gedurende enkele jaren maakten die tools voor het automatisch bewijzen van terminatie van termherschrijfsystemen enorme ontwikkelingen door. Vanaf 2008 waren die tools min of meer uitontwikkeld, en kwamen er niet veel nieuwe technieken meer bij. De mensen in dit vakgebied verlegden hun aandacht naar andere, vaak verwante onderwerpen.*

*Research in term rewriting also evolved. In the 1990s, when developing techniques for proving termination, for example, the focus was purely on theory: you described a new technique, provided some examples, and could publish an article about it. Around 2003, the bar was raised considerably: tools were developed that could automatically provide termination proofs, and a database of rewriting systems on which you could apply these tools. And there was an annual competition, a contest where the tools' scores on the examples from that database were assessed. From that point on, you could only get away with a new technique if you implemented it in a tool, and that tool scored well in that competition. I also participated in this, limited to the category of string rewriting systems, and won a few times. For several years, the tools for automatically proving termination of term rewriting systems underwent enormous developments. By 2008, these tools had more or less reached their full development stage, and not many new technologies were being developed.*

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## ► Termination of Extended String Rewriting

► Termination of Extended String Rewriting

► w/ Claude Marché:

## The Termination Competition 2006

### The Termination Competition

Claude Marché<sup>1,2</sup> and Hans Zantema<sup>3</sup>

<sup>1</sup> INRIA Futurs, ProVal, Parc Orsay Université, F-91893

<sup>2</sup> Lab. de Recherche en Informatique, Univ Paris-Sud, CNRS, Orsay, F-91405

<sup>3</sup> Department of Computer Science, Technische Universiteit Eindhoven

P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

Claude.Marche@inria.fr, h.zantema@tue.nl

**Abstract.** Since 2004, a Termination Competition is organized every year. This competition boosted a lot the development of automatic termination tools, but also the design of new techniques for proving termination. We present the background, results, and conclusions of the three first editions, and discuss perspectives and challenges for the future.

#### 1 Motivation and History

In a landmark paper in 1970, Manna & Ness [1] proposed a criterion for proving termination of rewrite systems, based on *reduction* orderings. Since then, many techniques for proving termination have been proposed, by providing means of defining classes of reduction orderings: path orderings, polynomial interpretations, etc. A few implementations were developed [2,3,4] but practical results in proposed sets of benchmark problems [5,6] were quite poor.

A disruptive progress came up in 1997, with the *Dependency Pair* criteria proposed by Arts & Giesl [7], allowing to prove termination with a larger class of orderings than reduction orderings. This brought up a new interest towards automation of termination proofs. Since around 2000 several tools were devel-

# WST 2006

- ▶ Termination of Extended String Rewriting

- ▶ w/ Claude Marché:

  - The Termination Competition 2006

- ▶ Frédéric Blanqui

  - Higher-Order Dependency Pairs

## ► Termination of Extended String Rewriting

► w/ Claude Marché:

## The Termination Competition 2006

► Frédéric Blanqui

## Higher-Order Dependency Pairs

### PAPER

## Static Dependency Pair Method Based on Strong Computability for Higher-Order Rewrite Systems

Keiichirou KUSAKARI<sup>†a)</sup>, Member, Yasuo ISOGAI<sup>†b)</sup>, Nonmember, Masahiko SAKAI<sup>†c)</sup>, Member, and Frédéric BLANQUI<sup>††d)</sup>, Nonmember

**SUMMARY** Higher-order rewrite systems (HRSs) and simply-typed term rewriting systems (STRSs) are computational models of functional programs. We recently proposed an extremely powerful method, the static dependency pair method, which is based on the notion of strong computability, in order to prove termination in STRSs. In this paper, we extend the method to HRSs. Since HRSs include  $\lambda$ -abstraction but STRSs do not, we restructure the static dependency pair method to allow  $\lambda$ -abstraction, and show that the static dependency pair method also works well on HRSs without new restrictions.

**key words:** higher-order rewrite system, termination, static dependency pair, plain function-passing, strong computability, subterm criterion

### 1. Introduction

A term rewriting system (TRS) is a computational model that provides operational semantics for functional programs [22]. A TRS cannot, however, directly handle higher-order functions, which are widely used in functional programming languages. Simply-typed term rewriting systems (STRSs) [12] and higher-order rewrite systems (HRSs) [17] have been introduced to extend TRSs. These rewriting systems can directly handle higher-order functions. For example, a typical higher-order function `foldl` can be represented by the following HRS  $R_{\text{foldl}}$ :

$$\begin{cases} \text{foldl}(\lambda xy.F(x,y), X, \text{nil}) \rightarrow X \\ \text{foldl}(\lambda xy.F(x,y), X, \text{cons}(Y,L)) \\ \quad \rightarrow \text{foldl}(\lambda xy.F(x,y), F(X,Y), L) \end{cases}$$

HRSs can represent anonymous functions because HRSs have a  $\lambda$ -abstraction syntax, which STRSs do not. For instance, an anonymous function  $\lambda xy.\text{add}(x.\text{mul}(y,y))$  is used in the HRS  $R_{\text{sum}}$ .

Here, the function `sqsum` returns the square sum  $x_1^2 + x_2^2 + \dots + x_n^2$  from an input list  $[x_1, x_2, \dots, x_n]$ .

As a method for proving termination of TRSs, Arts and Giesl proposed the dependency pair method for TRSs based on recursive structure analysis [1], which was then extended to STRSs [12], and to HRSs [18].

In higher-order settings, there are two kinds of analysis for recursive structures. One is dynamic analysis, and the other is static analysis. The extensions in [12] and [18] analyze dynamic recursive structures based on function-call dependency relationships, but not on relationships that may be extracted syntactically from function definitions. When a program runs, some functions can be substituted for higher-order variables. Dynamic recursive structure analysis considers dependencies through higher-order variables. Static recursive structure analysis on the other hand, does not consider such dependencies.

For example, consider the HRS  $R_{\text{sqsum}}$ . The dynamic dependency pair method in [18] extracts the following 9 pairs, called dynamic dependency pairs:

$$\begin{cases} \text{foldl}^1(\lambda xy.F(x,y), X, \text{cons}(Y,L)) \\ \quad \rightarrow \text{foldl}^1(\lambda xy.F(x,y), F(X,Y), L) & (a) \\ \text{foldl}^2(\lambda xy.F(x,y), X, \text{cons}(Y,L)) \rightarrow F(c_x, c_y) & (b) \\ \text{foldl}^3(\lambda xy.F(x,y), X, \text{cons}(Y,L)) \rightarrow F(X, Y) & (c) \\ \text{add}^1(s(X), Y) \rightarrow \text{add}^2(X, Y) & (d) \\ \text{mul}^2(s(X), Y) \rightarrow \text{add}^2(\text{mul}(X, Y), Y) & (e) \\ \text{mul}^3(s(X), Y) \rightarrow \text{mul}^2(X, Y) & (f) \\ \text{sqsum}^1(l) \rightarrow \text{foldl}^4(\lambda xy.\text{add}(x.\text{mul}(y,y)), 0, L) & (g) \end{cases}$$



# WST 2006

- ▶ Termination of Extended String Rewriting

- ▶ w/ Claude Marché:

  - The Termination Competition 2006

- ▶ Frédéric Blanqui

  - Higher-Order Dependency Pairs

- ▶ Naoki Nishida

  - Dependency Graph Method for Proving  
Termination of Narrowing

## ► Termination of Extended String Rewriting

## ► w/ Claude Marché:

## The Termination Competition 2006

## ► Frédéric Blanqui

## Higher-Order Dependency Pairs

## ► Naoki Nishida

## Dependency Graph Method for Proving Termination of Narrowing

## Goal-Directed and Relative Dependency Pairs for Proving the Termination of Narrowing\*

José Iborra<sup>1</sup>, Naoki Nishida<sup>2</sup>, and Germán Vidal<sup>1</sup>

<sup>1</sup> DSIC, Universidad Politécnica de Valencia, Spain  
{jiborra,gvidal}@dsic.upv.es

<sup>2</sup> Graduate School of Information Science, Nagoya University, Nagoya, Japan  
nishida@is.nagoya-u.ac.jp

**Abstract.** In this work, we first consider a *goal-oriented* extension of the dependency pair framework for proving termination w.r.t. a given set of initial terms. Then, we introduce a new result for proving *relative* termination in terms of a dependency pair problem. Both contributions put together allow us to define a simple and powerful approach to analyzing the termination of *narrowing*, an extension of rewriting that replaces matching with unification in order to deal with logic variables. Our approach could also be useful in other contexts where considering termination w.r.t. a given set of terms is also natural (e.g., proving the termination of functional programs).

## 1 Introduction

Proving that a program terminates is a fundamental problem that has been extensively studied in almost all programming paradigms. In *term rewriting*, where termination analysis is a considerable challenge (see, e.g., [1]),

# WST 2006

- ▶ Termination of Extended String Rewriting
- ▶ w/ Claude Marché:  
The Termination Competition 2006
- ▶ Frédéric Blanqui  
Higher-Order Dependency Pairs
- ▶ Naoki Nishida  
Dependency Graph Method for Proving  
Termination of Narrowing



- ▶ Termination of Extended String Rewriting

- ▶ w/ Claude Marché:

  - The Termination Competition 2006

- ▶ Frédéric Blanqui

  - Higher-Order Dependency Pairs

- ▶ Naoki Nishida

  - Dependency Graph Method for Proving  
Termination of Narrowing

- ▶ Frédéric Blanqui et al

  - CoLoR: A Coq Library on Rewriting and Termination

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ w/ Adam Koprowski:

## Certification of Matrix Interpretations in Coq

## Certification of Matrix Interpretations in Coq

## Certification of Proving Termination of Term Rewriting by Matrix Interpretations\*

Adam Koprowski and Hans Zantema

Eindhoven University of Technology  
 Department of Computer Science  
 P.O. Box 513, 5600 MB, Eindhoven, The Netherlands  
 {A.Koprowski,H.Zantema}@tue.nl

**Abstract.** We develop a Coq formalization of the matrix interpretation method, which is a recently developed, powerful approach to proving termination of term rewriting. Our formalization is a contribution to the CoLoR project and allows to automatically certify matrix interpretation proofs produced by tools for proving termination. Thanks to this development the combination of CoLoR and our tool, TPA, was the winner in 2007 in the new certified category of the annual Termination Competition.

## 1 Introduction

Termination is an important concept in term rewriting. Many methods for proving termination have been proposed over the years. Recently the emphasis in this area is on automation and a number of tools have been developed for that purpose. One of such tools is TPA [13] developed by the first author.

To evaluate termination tools and stimulate their improvement the annual Termination Competition [3] is organized, where such tools compete on a set of problems from the Termination Problems Database (TPDB), [4]. This competition has become a de facto standard in evaluation of new termination techniques.

# WST 2007

- ▶ w/ Adam Koprowski:

Certification of Matrix Interpretations in Coq

- ▶ w/ Claude Marché, Johannes Waldmann:

The Termination Competition 2007



# WST 2007

- ▶ w/ Adam Koprowski:  
Certification of Matrix Interpretations in Coq
- ▶ w/ Claude Marché, Johannes Waldmann:  
The Termination Competition 2007
- ▶ Johannes Waldmann  
Arctic Termination

- ▶ w/ Adam Koprowski:

## Certification of Matrix Interpretations in Coq

- ▶ w/ Claude Marché, Johannes Waldmann:

## The Termination Competition 2007

- ▶ Johannes Waldmann

## Arctic Termination

### Arctic Termination ... Below Zero

Adam Koprowski<sup>1</sup> and Johannes Waldmann<sup>2</sup>

<sup>1</sup> Department of Computer Science

Eindhoven University of Technology

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

<sup>2</sup> Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig

Fb IMN, PF 30 11 66, D-04251 Leipzig, Germany

**Abstract.** We introduce the arctic matrix method for automatically proving termination of term rewriting. We use vectors and matrices over the arctic semi-ring: natural numbers extended with  $-\infty$ , with the operations “max” and “plus”. This extends the matrix method for term rewriting and the arctic matrix method for string rewriting. In combination with the Dependency Pairs transformation, this allows for some conceptually simple termination proofs in cases where only much more involved proofs were known before. We further generalize to arctic numbers “below zero”: integers extended with  $-\infty$ . This allows to treat some termination problems with symbols that require a predecessor semantics. The contents of the paper has been formally verified in the Coq proof assistant and the formalization has been contributed to the CoLoR library of certified termination techniques. This allows formal verification of termination proofs using the arctic matrix method. We also report on experiments with an implementation of this method which, compared to results from 2007, outperforms TPA (winner of the certified termination

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- Combinatorial Problems by Termination of Rewriting

## ► Combinatorial Problems by Termination of Rewriting

### Example

SRS     $h11 \rightarrow 1h$      $1s \rightarrow s1$      $bt \rightarrow bh$   
       $11hb \rightarrow 11sb$      $1t \rightarrow t111$   
       $h1b \rightarrow t11b$      $bs \rightarrow bh$

models Collatz problem

## ► Combinatorial Problems by Termination of Rewriting

### Example

SRS     $h11 \rightarrow 1h$      $1s \rightarrow s1$      $bt \rightarrow bh$   
        $11hb \rightarrow 11sb$      $1t \rightarrow t111$   
        $h1b \rightarrow t11b$      $bs \rightarrow bh$

models Collatz problem



## An Automated Approach to the Collatz Conjecture

Emre Yolcu<sup>1</sup> · Scott Aaronson<sup>2</sup> · Marijn J. H. Heule<sup>1</sup>

Received: 3 December 2021 / Accepted: 30 December 2022 / Published online: 25 April 2023  
 © The Author(s) 2023

### Abstract

We explore the Collatz conjecture and its variants through the lens of termination of string rewriting. We construct a rewriting system that simulates the iterated application of the Collatz function on strings corresponding to mixed binary–ternary representations of positive integers. We prove that the termination of this rewriting system is equivalent to the Collatz conjecture. We also prove that a previously studied rewriting system that simulates the Collatz function using unary representations does not admit termination proofs via natural matrix interpretations, even when used in conjunction with dependency pairs. To show the feasibility of our approach in proving mathematically interesting statements, we implement a minimal termination prover that uses natural/arctic matrix interpretations and we find automated proofs of nontrivial weakenings of the Collatz conjecture. Although we do not succeed in proving the Collatz conjecture, we believe that the ideas here represent an interesting new approach.

**Keywords** String rewriting · Termination · Matrix interpretations · SAT solving · Collatz conjecture · Computer-assisted mathematics

## ► Combinatorial Problems by Termination of Rewriting

### Example

SRS     $h11 \rightarrow 1h$      $1s \rightarrow s1$      $bt \rightarrow bh$   
       $11hb \rightarrow 11sb$      $1t \rightarrow t111$   
       $h1b \rightarrow t11b$      $bs \rightarrow bh$

models Collatz problem

## ► Karel Chvalovský

Derivational Complexity of

$\{aa \rightarrow bc, bb \rightarrow ac, cc \rightarrow ab\}$

## ► Combinatorial Problems by Termination of Rewriting

## ► Karel Chvalovský

### Derivational Complexity of

$$\{aa \rightarrow bc, bb \rightarrow ac, cc \rightarrow ab\}$$

### On a Method for Proving Exact Bounds on Derivational Complexity in Thue Systems

S. I. Adian\*

*Steklov Mathematical Institute, Russian Academy of Sciences*

Received January 9, 2012

**Abstract**—In this paper, the following system of substitutions in a 3-letter alphabet

$$\Sigma = \langle a, b, c \mid a^2 \rightarrow bc, b^2 \rightarrow ac, c^2 \rightarrow ab \rangle$$

is considered. A detailed proof of results that were described briefly in the author's paper [1] is presented. They give an answer to the specific question on the possibility of giving a polynomial upper bound for the lengths of derivations from a given word in the system  $\Sigma$  stated in the literature. The maximal possible number of steps in derivation sequences starting from a given word  $W$  is denoted by  $\mathbf{D}(W)$ . The maximum of  $\mathbf{D}(W)$  for all words of length  $|W| = l$  is denoted by  $\mathbf{D}(l)$ . It is proved that the function  $\mathbf{D}(W)$  on words  $W$  of given length  $|W| = m + 2$  reaches its maximum only on words of the form  $W = c^2 b^m$  and  $W = b^m a^2$ . For these words, the following precise estimate is established:

$$\mathbf{D}(m + 2) = \mathbf{D}(c^2 b^m) = \mathbf{D}(b^m a^2) = \left\lceil \frac{3m^2}{2} \right\rceil + m + 1 < \frac{3(m + 1)^2}{2},$$

where  $\lceil 3m^2/2 \rceil$  for odd  $|m|$  is the round-up of  $3m^2/2$  to the nearest integer.

**DOI:** 10.1134/S0001434612070012

**Keywords:** word rewriting system, derivational complexity, Thue system, polynomial upper bound, left (right) divisibility of a word.

In the author's paper [1], a quadratic upper bound was found for the maximal length of derivation sequences starting from a given word  $W$  in the word rewriting system  $\Sigma$  presented by the following substitution rules:



## ► Combinatorial Problems by Termination of Rewriting

---

<sup>1</sup>Zantema mentioned in [5] that his system is a simplification of an original system with the same properties that was presented by D. Hofbauer and J. Waldmann in their talk at the meeting “Proof Theory and Rewriting” in Obergurgl, March 30, 2010. Their system contained four derivation rules. At the end of the previous century, some authors started to call Thue systems “Word Rewriting Systems” (shortly, WRS-systems). More recently, some other authors, working in computer science, are attempting to change this to “SRS-systems,” because, for some unknown reason, they are now using (without any comment) the term “*string*” instead of the classical term “*word*.” We do not think that this practice is reasonable.

MATHEMATICAL NOTES Vol. 92 No. 1 2012

## ► Combinatorial Problems by Termination of Rewriting

## ► Karel Chvalovský

### Derivational Complexity of

$$\{aa \rightarrow bc, bb \rightarrow ac, cc \rightarrow ab\}$$



Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

Information Processing Letters 98 (2006) 156–158

IPL 2006

Information  
Processing  
Letters

[www.elsevier.com/locate/ipl](http://www.elsevier.com/locate/ipl)

## Termination of $\{aa \rightarrow bc, bb \rightarrow ac, cc \rightarrow ab\}$

Dieter Hofbauer<sup>a,\*</sup>, Johannes Waldmann<sup>b</sup>

<sup>a</sup> Mühlegasse 16, D-34125 Kassel, Germany

<sup>b</sup> Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fb IMN, PF 301166, D-04251 Leipzig, Germany

Received 29 September 2005; received in revised form 20 November 2005; accepted 22 December 2005

Available online 30 January 2006

Communicated by D. Basin

**Keywords:** Automatic theorem proving; Termination; String rewriting

### 1. The problem

“Start by a finite string over the alphabet  $\{a, b, c\}$ . As long as two consecutive symbols are the same, they may be replaced by the other two symbols in alphabetic order. [...] Can this go on forever?” This problem, i.e., termination of the string rewriting system

$$R = \{aa \rightarrow bc, bb \rightarrow ac, cc \rightarrow ab\}$$

was posed by Zanema [9] as Problem #104 in the RTA list of open problems [3,4]. The following is an example rewriting sequence (redexes underlined):

participating programs nor by their authors, and to our knowledge it remained open until the solution given below was found.

We find the problem attractive due to its seeming simplicity. Firstly, the system  $R$  is highly symmetric as a renaming of its own reverse. Secondly, it is small: 3 rules with 12 letters. Note however that (uniform) termination is undecidable even for string rewriting systems with three rules [6]. Finally,  $R$  is length-preserving. However, also for length-preserving string rewriting systems the termination problem remains undecidable [2].

## ► Combinatorial Problems by Termination of Rewriting

J Autom Reasoning (2008) 40:195–220  
DOI 10.1007/s10817-007-9087-9

### Matrix Interpretations for Proving Termination of Term Rewriting

Jörg Endrullis · Johannes Waldmann · Hans Zantema

Received: 10 October 2007 / Accepted: 12 October 2007 / Published online: 11 December 2007  
© Springer Science + Business Media B.V. 2007

**Abstract** We present a new method for automatically proving termination of term rewriting. It is based on the well-known idea of interpretation of terms where every rewrite step causes a decrease, but instead of the usual natural numbers we use vectors of natural numbers, ordered by a particular nontotal well-founded ordering. Function symbols are interpreted by linear mappings represented by matrices. This method allows us to prove termination and relative termination. A modification of the latter, in which strict steps are only allowed at the top, turns out to be helpful in combination with the dependency pair transformation. By bounding the dimension and the matrix coefficients, the search problem becomes finite. Our implementation transforms it to a Boolean satisfiability problem (SAT), to be solved by a state-of-the-art SAT solver.

- Combinatorial Problems by Termination of Rewriting
- Pierre Courtieu, Gladys Gbedo, Olivier Pons  
Matrix Interpretations Revisited

## Improved Matrix Interpretation\*

Pierre Courtieu, Gladys Gbedo, and Olivier Pons

CÉDRIC – CNAM, Paris, France

**Abstract.** We present a new technique to prove termination of Term Rewriting Systems, with full automation. A crucial task in this context is to find suitable well-founded orderings. A popular approach consists in interpreting terms into a domain equipped with an adequate well-founded ordering. In addition to the usual interpretations: natural numbers or polynomials over integer/rational numbers, the recently introduced matrix based interpretations have proved to be very efficient regarding termination of string rewriting and of term rewriting. In this spirit we propose to interpret terms as polynomials over integer matrices. Designed for term rewriting, our generalisation subsumes previous approaches allowing for more orderings without increasing the search space. Thus it performs better than the original version. Another advantage is that, interpreting terms to actual polynomials of matrices, it opens the way to matrix non linear interpretations. This result is implemented in the CiME3 rewriting toolkit.

### 1 Introduction

The property of termination, well-known to be undecidable, is fundamental in many aspects of computer science and logic. It is crucial in the proof of programs correctness under deductive reasoning. Despite its non decidability, many

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Length Preserving String Rewriting with Exponential Derivation Length

- ▶ Length Preserving String Rewriting with Exponential Derivation Length

## Example

SRS  $0b \rightarrow cb$   $0c \rightarrow 10$   $1c \rightarrow c0$

has exponential derivation length

- ▶ Length Preserving String Rewriting with Exponential Derivation Length

## Example

SRS  $0b \rightarrow cb \quad 0c \rightarrow 10 \quad 1c \rightarrow c0$

has exponential derivation length

- ▶ Dieter Hofbauer, Johannes Waldmann  
Match-Bounds for Relative Termination



- ▶ Length Preserving String Rewriting with Exponential Derivation Length

## Example

SRS  $0b \rightarrow cb \quad 0c \rightarrow 10 \quad 1c \rightarrow c0$

has exponential derivation length

- ▶ Dieter Hofbauer, Johannes Waldmann  
Match-Bounds for Relative Termination
- ▶ Freidrich Neurauter, Aart Middeldorp  
Polynomial Termination over  $\mathbb{N}$  and  $\mathbb{R}$  does not imply Polynomial Termination over  $\mathbb{Q}$

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Christian Sternagel, René Thiemann  
A Relative Dependency Pair Framework

- ▶ Christian Sternagel, René Thiemann  
A Relative Dependency Pair Framework

## Relative Termination via Dependency Pairs

José Iborra<sup>1</sup> · Naoki Nishida<sup>2</sup> · Germán Vidal<sup>3</sup> ·  
Akihisa Yamada<sup>4</sup> 

Received: 4 April 2016 / Accepted: 8 April 2016 / Published online: 30 April 2016  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** A term rewrite system is terminating when no infinite reduction sequences are possible. Relative termination generalizes termination by permitting infinite reductions as long as some distinguished rules are not applied infinitely many times. Relative termination is thus a fundamental notion that has been used in a number of different contexts, like analyzing the confluence of rewrite systems or the termination of narrowing. In this work, we introduce a novel technique to prove relative termination by reducing it to dependency pair problems. To the best of our knowledge, this is the first significant contribution to Problem #106 of the RTA List of Open Problems. We first present a general approach that is then instantiated to provide a concrete technique for proving relative termination. The practical significance of our method is illustrated by means of an experimental evaluation.

- ▶ Christian Sternagel, René Thiemann  
A Relative Dependency Pair Framework



## A Dependency Pair Framework for Relative Termination of Term Rewriting

Jan-Christoph Kassing<sup>(✉)</sup>, Grigory Vartanyan<sup>(iD)</sup>, and Jürgen Giesl<sup>(iD)</sup>

RWTH Aachen University, Aachen, Germany  
{kassing,giesl}@cs.rwth-aachen.de, grigory.vartanyan@rwth-aachen.de

**Abstract.** *Dependency pairs* are one of the most powerful techniques for proving termination of term rewrite systems (TRSs), and they are used in almost all tools for termination analysis of TRSs. Problem #106 of the RTA List of Open Problems asks for an adaption of dependency pairs for *relative termination*. Here, infinite rewrite sequences are allowed, but one wants to prove that a certain subset of the rewrite rules cannot be used infinitely often. Dependency pairs were recently adapted to *annotated dependency pairs (ADPs)* to prove almost-sure termination of probabilistic TRSs. In this paper, we develop a novel adaption of ADPs for relative termination. We implemented our new ADP framework in our tool AProVE and evaluate it in comparison to state-of-the-art tools for relative termination of TRSs.

### 1 Introduction

Termination is an important property in verification. There is a wealth of

- ▶ Christian Sternagel, René Thiemann  
A Relative Dependency Pair Framework



## A Dependency Pair Framework for Relative Termination of Term Rewriting

Jan-Christoph Kassing<sup>(✉)</sup>, Grigory Vartanyan<sup>(iD)</sup>, and Jürgen Giesl<sup>(iD)</sup>

RWTH Aachen University, Aachen, Germany  
{kassing,giesl}@cs.rwth-aachen.de, grigory.vartanyan@rwth-aachen.de

**Abstract.** *Dependency pairs* are one of the most powerful techniques for proving termination of term rewrite systems (TRSs), and they are used in almost all tools for termination analysis of TRSs. Problem #106 of the RTA List of Open Problems asks for an adaption of dependency pairs for *relative termination*. Here, infinite rewrite sequences are allowed, but one wants to prove that a certain subset of the rewrite rules cannot be used infinitely often. Dependency pairs were recently adapted to *annotated dependency pairs (ADPs)* to prove almost-sure termination of probabilistic TRSs. In this paper, we develop a novel adaption of ADPs for relative termination. We implemented our new ADP framework in our tool AProVE and evaluate it in comparison to state-of-the-art tools for relative termination of TRSs.

### 1 Introduction

Termination is an important property for program verification. There is a wealth of

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

## ► Automatically Finding Non-Confluent Examples in Abstract Rewriting

*Math. Struct. in Comp. Science* (2018), vol. 28, pp. 1485–1505. © Cambridge University Press 2018  
doi:10.1017/S0960129518000221 First published online 25 July 2018

### Finding small counterexamples for abstract rewriting properties

HANS ZANTEMA

*Department of Computer Science, TU Eindhoven, P.O. Box 513,  
5600 MB Eindhoven, The Netherlands.*

*Email: H.Zantema@tue.nl, and*

*Institute for Computing and Information Sciences, Radboud University  
Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands*

*Received 1 February 2012; revised 1 November 2017*

Rewriting notions like termination, normal forms and confluence can be described in an abstract way referring to rewriting only as a binary relation. Several theorems on rewriting, like Newman's lemma, can be proved in this abstract setting. For investigating possible generalizations of such theorems, it is fruitful to have counterexamples showing that particular generalizations do not hold. In this paper, we develop a technique to find such counterexamples fully automatically, and we describe our tool *Carpa* that follows this technique. The basic idea is to fix the number of objects of the abstract rewrite system, and to express the conditions and the negation of the conclusion in a satisfiability (SAT) formula, and then call a current SAT solver. In case the formula turns out to be satisfiable, the resulting satisfying assignment yields a counterexample to the encoded property. We give several examples of finite abstract rewrite systems having remarkable properties that are found in this way fully automatically.

#### 1. Introduction



## ► Automatically Finding Non-Confluent Examples in Abstract Rewriting

### A REWRITING VIEW OF SIMPLE TYPING

AARON STUMP<sup>a</sup>, HANS ZANTEMA<sup>b</sup>, GARRIN KIMMELL<sup>c</sup>, AND RUBA EL HAJ OMAR

<sup>a,c,d</sup> Computer Science, The University of Iowa  
*e-mail address:* astump@acm.org, {garrin-kimmell,roba-elhajomar}@uiowa.edu

<sup>b</sup> Department of Computer Science, TU Eindhoven, The Netherlands; and Institute for Computing and Information Sciences, Radboud University, The Netherlands  
*e-mail address:* h.zantema@tue.nl

**ABSTRACT.** This paper shows how a recently developed view of typing as small-step abstract reduction, due to Kuan, MacQueen, and Findler, can be used to recast the development of simple type theory from a rewriting perspective. We show how standard meta-theoretic results can be proved in a completely new way, using the rewriting view of simple typing. These meta-theoretic results include standard type preservation and progress properties for simply typed lambda calculus, as well as generalized versions where typing is taken to include both abstract and concrete reduction. We show how automated analysis tools developed in the term-rewriting community can be used to help automate the proofs for this meta-theory. Finally, we show how to adapt a standard proof of normalization of simply typed lambda calculus, for the rewriting approach to typing.

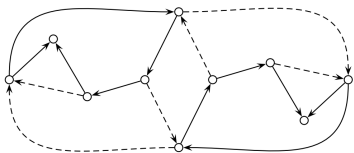
#### 1. INTRODUCTION

This paper develops a significant part of the theory of simple types based on a recently introduced rewriting approach to typing. The idea of viewing typing as a small-step abstract reduction relation was proposed by Kuan, MacQueen, and Findler in 2007, and explored also by Ellison, Șerbănuță, and Roșu [13, 9, 14]. These works sought to use rewrite systems to specify typing as first-order logic. Our motivation is to

## ► Automatically Finding Non-Confluent Examples in Abstract Rewriting

12

A. STUMP, H. ZANTEMA, G. KIMMELL, AND R. EL HAJ OMAR



In this example there are two convertible normal forms, so the union is not confluent, and both  $\rightarrow_a$  and  $\rightarrow_b$  are both confluent and terminating;  $\rightarrow_a$  is even deterministic. Also condition (3) of Theorem 5.3 is easily checked, even stronger:  $\leftarrow_a \cdot \rightarrow_b \subseteq \rightarrow_{ba} \cdot \leftarrow_a$ . This example was found using a SAT solver. A direct encoding of the example to be looked for run out of resources. However, by adding a symmetry requirement, was observed on the first example, the SAT solver yielded a satisfying assignment that could be interpreted as a valid example. The example given above was obtained from this after removing some redundant arrows. Independently, Bertram Felgenhauer found an example that could be simplified to exactly the same example as given here. This remarkable example was the starting point of developing the tool CARPA by which such examples can be found fully automatically.

### A REWRITING VIEW OF SIMPLE TYPING

AARON STUMP<sup>a</sup>, HANS ZANTEMA<sup>b</sup>, GARRIN KIMMELL<sup>c</sup>, AND RUBA EL HAJ OMAR<sup>a,c,d</sup> Computer Science, The University of Iowa

e-mail address: astump@acm.org, {garrin-kimmell,roba-elhajomar}@uiowa.edu

<sup>b</sup> Department of Computer Science, TU Eindhoven, The Netherlands; and Institute for Computing and Information Sciences, Radboud University, The Netherlands

e-mail address: h.zantema@tue.nl

**ABSTRACT.** This paper shows how a recently developed view of typing as small-step abstract reduction, due to Kuan, MacQueen, and Findler, can be used to recast the development of simple type theory from a rewriting perspective. We show how standard meta-theoretic results can be proved in a completely new way, using the rewriting view of simple typing. These meta-theoretic results include standard type preservation and progress properties for simply typed lambda calculus, as well as generalized versions where typing is taken to include both abstract and concrete reduction. We show how automated analysis tools developed in the term-rewriting community can be used to help automate the proofs for this meta-theory. Finally, we show how to adapt a standard proof of normalization of simply typed lambda calculus, for the rewriting approach to typing.

### 1. INTRODUCTION

This paper develops a significant part of the theory of simple types based on a recently introduced rewriting approach to typing. The idea of viewing typing as a small-step abstract reduction relation was proposed by Kuan, MacQueen, and Findler in 2007, and explored also by Ellison, Șerbănuță, and Roșu [13, 9, 14]. These works sought to use rewrite systems to specify typing as a first-order property. Our motivation is to

- ▶ Bertram Felgenhauer

## A Proof Order for Decreasing Diagrams

► Bertram Felgenhauer

## A Proof Order for Decreasing Diagrams

### Proof Orders for Decreasing Diagrams

Bertram Felgenhauer<sup>1</sup> and Vincent van Oostrom<sup>2</sup>

<sup>1</sup> Institute for Computer Science, University of Innsbruck  
bertram.felgenhauer@uibk.ac.at

<sup>2</sup> Department of Philosophy, Utrecht University  
Vincent.vanOostrom@phil.uu.nl

#### Abstract

We present and compare some well-founded proof orders for decreasing diagrams. These proof orders order a conversion above another conversion if the latter is obtained by filling any peak in the former by a (locally) decreasing diagram. Therefore each such proof order entails the decreasing diagrams technique for proving confluence. The proof orders differ with respect to monotonicity and complexity. Our results are developed in the setting of involutive monoids. We extend these results to obtain a decreasing diagrams technique for confluence modulo.

**1998 ACM Subject Classification** F.4 Mathematical Logic and Formal Languages

**Keywords and phrases** involutive monoid, confluence modulo, decreasing diagram, proof order

**Digital Object Identifier** 10.4230/LIPIcs.RTA.2013.174

#### 1 Introduction

In this paper we revisit the decreasing diagrams technique [10] for proving confluence. We exhibit several well-founded orders on proofs that allow us to prove termination of the proof transformation system defined by the locally decreasing diagrams. A similar approach is used in the correctness proof for completion by Bachmair and Dershowitz [2]. Rather than working on proofs directly we develop our orders in the setting of involutive monoids, which

# IWC 2012

- ▶ Bertram Felgenhauer

A Proof Order for Decreasing Diagrams



four tool presentations (ACP, CSI, Saigawa, CeTA)

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Automatically Finding Non-Confluent Examples in Term Rewriting

► Automatically Finding Non-Confluent Examples in Term Rewriting



## Automating the First-Order Theory of Rewriting for Left-Linear Right-Ground Rewrite Systems\*

Franziska Rapp<sup>1</sup> and Aart Middeldorp<sup>2</sup>

- 1 Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
franziska.rapp@uibk.ac.at
- 2 Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
aart.middeldorp@uibk.ac.at

### Abstract

The first-order theory of rewriting is decidable for finite left-linear right-ground rewrite systems. We present a new tool that implements the decision procedure for this theory. It is based on tree automata techniques. The tool offers the possibility to synthesize rewrite systems that satisfy properties that are expressible in the first-order theory of rewriting.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases first-order theory, ground rewrite systems, automation, synthesis

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.36

### 1 Introduction

Dauchet and Tison [4] proved that the first-order theory of rewriting is decidable for ground rewrite systems. In this theory one can express properties like confluence

$$\forall s \forall t \forall u (s \rightarrow^* t \wedge s \rightarrow^* u \implies \exists v (t \rightarrow^* v \wedge u \rightarrow^* v))$$

and normalization

$$\forall s \exists t (s \rightarrow^* t \wedge \neg \exists u (s \rightarrow^* u))$$



► Automatically Finding Non-Confluent Examples in Term Rewriting

Journal of Automated Reasoning (2023) 67:14  
<https://doi.org/10.1007/s10817-023-09661-7>

## RESEARCH



## First-Order Theory of Rewriting for Linear Variable-Separated Rewrite Systems: Automation, Formalization, Certification

Aart Middeldorp<sup>1</sup> · Alexander Lochmann<sup>1</sup> · Fabian Mitterwallner<sup>1</sup>

Received: 22 April 2022 / Accepted: 19 January 2023 / Published online: 6 April 2023  
© The Author(s) 2023

### Abstract

The first-order theory of rewriting is decidable for linear variable-separated rewrite systems. We present a new decision procedure which is the basis of FORT, a decision and synthesis tool for properties expressible in the theory. The decision procedure is based on tree automata techniques and verified in Isabelle. Several extensions make the theory more expressive and FORT more versatile. We present a certificate language that enables the output of FORT to be certified by the certifier FORTify generated from the formalization, and we provide extensive experiments.

**Keywords** Term rewriting · First-order theory · Tree automata · Formalization

### 1 Introduction

Many properties of rewrite systems can be expressed as logical formulas in the first-order

- ▶ Automatically Finding Non-Confluent Examples in Term Rewriting

- ▶ Takahito Aoto

## Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering

### Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering

Takahito Aoto

RIEC, Tohoku University,  
2-1-1 Katahira, Aoba-ku, Sendai 980-8577, Japan  
aoto@nue.riec.tohoku.ac.jp

**Abstract.** In order to disprove confluence of term rewriting systems, we develop new criteria for ensuring non-joinability of terms based on interpretation and ordering. We present some instances of the criteria which are amenable for automation, and report on an implementation of a confluence disproving procedure based on these instances. The experiments reveal that our method is successfully applied to automatically disprove confluence of some term rewriting systems, on which state-of-the-art automated confluence provers fail. A key idea to make our method effective is the introduction of usable rules—this allows one to decompose the constraint on rewrite rules into smaller components that depend on starting terms.

**Keywords:** Confluence, Non-Joinability, Interpretation, Ordering, Term Rewriting Systems.

1 Introduction

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Nachum Dershowitz

Dependency Pairs are a Simple Semantic  
Path Order

- ▶ Nachum Dershowitz

Dependency Pairs are a Simple Semantic  
Path Order

- ▶ Cynthia Kop

Termination of LCTRSs

# WST 2013

- ▶ Nachum Dershowitz

Dependency Pairs are a Simple Semantic  
Path Order

- ▶ Cynthia Kop

Termination of LCTRSs



- ▶ Nachum Dershowitz

Dependency Pairs are a Simple Semantic  
Path Order

- ▶ Cynthia Kop

Termination of LCTRSs

- ▶ Akihisa Yamada, Keiichirou Kusakari

Toshiki Sakabe

Partial Status for KBO

## ► Nachum Dershowitz

Dependency Pairs are a Simple Semantic Path Order

## ► Cynthia Kop

Termination of LCTRSs

## ► Akihisa Yamada, Keiichirou Kusakari Toshiki Sakabe

Partial Status for KBO



Contents lists available at ScienceDirect

Science of Computer Programming

[www.elsevier.com/locate/scico](http://www.elsevier.com/locate/scico)



## A unified ordering for termination proving

Akihisa Yamada<sup>a,b,\*</sup>, Keiichirou Kusakari<sup>c</sup>, Toshiki Sakabe<sup>a</sup>

<sup>a</sup> Graduate School of Information Science, Nagoya University, Japan

<sup>b</sup> National Institute of Advanced Industrial Science and Technology, Japan

<sup>c</sup> Faculty of Engineering, Gifu University, Japan



### ARTICLE INFO

#### Article history:

Received 11 December 2013

Received in revised form 7 July 2014

Accepted 9 July 2014

Available online 14 August 2014

#### Keywords:

Term rewriting

Reduction order

Termination

### ABSTRACT

We introduce a reduction order called the weighted path order (WPO) that subsumes many existing reduction orders. WPO compares weights of terms as in the Knuth–Bendix order (KBO), while WPO allows weights to be computed by a wide class of interpretations. We investigate summations, polynomials and maximums for such interpretations. We show that KBO is a restricted case of WPO induced by summations, the polynomial order (PLO) is subsumed by WPO induced by polynomials, and the lexicographic path order (LPO) is a restricted case of WPO induced by maximums. By combining these interpretations, we obtain an instance of WPO that unifies KBO, LPO and PLO. In order to fit WPO in the modern dependency pair framework, we further provide a reduction pair based on WPO and partial statuses. As a reduction pair, WPO also subsumes matrix interpretations. We finally present SMT encodings of our techniques, and demonstrate the significance of our work through experiments.

© 2014 Elsevier B.V. All rights reserved.

### 1. Introduction

Proving termination of term rewrite systems (TRSs) is one of the most important tasks in program verification and automated theorem proving, where reduction orders play a fundamental role. The classic use of reduction orders in termination proving is illustrated in the following example:



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Michio Oyamaguchi, Nao Hirokawa  
Confluence and Critical-Pair-Closing  
Systems



- Michio Oyamaguchi, Nao Hirokawa  
Confluence and Critical-Pair-Closing  
Systems

## Confluence by Critical Pair Analysis Revisited

Nao Hirokawa<sup>1</sup>✉, Julian Nagele<sup>2</sup>, Vincent van Oostrom<sup>3</sup>,  
and Michio Oyamaguchi<sup>4</sup>

<sup>1</sup> JAIST, Nomi, Japan

hirokawa@jaist.ac.jp

<sup>2</sup> Queen Mary University of London, London, UK

j.nagele@qmul.ac.uk

<sup>3</sup> University of Innsbruck, Innsbruck, Austria

Vincent.van-Oostrom@uibk.ac.at

<sup>4</sup> Nagoya University, Nagoya, Japan

oyamaguchi@za.ztv.ne.jp

**Abstract.** We present two methods for proving confluence of left-linear term rewrite systems. One is *hot-decreasingness*, combining the parallel/development closedness theorems with rule labelling based on a terminating subsystem. The other is *critical-pair-closing system*, allowing to boil down the confluence problem to confluence of a special subsystem whose duplicating rules are relatively terminating.

**Keywords:** Term rewriting · Confluence · Decreasing diagrams

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ w/ Jörg Endrullis:

## Non-Termination using Regular Languages

## Non-Termination using Regular Languages

## Proving non-termination by finite automata

Jörg Endrullis<sup>1</sup> and Hans Zantema<sup>2,3</sup>

- 1 Department of Computer Science, VU University Amsterdam, 1081 HV Amsterdam, The Netherlands, email: j.endrullis@vu.nl
- 2 Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, email: H.Zantema@tue.nl
- 3 Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

## Abstract

A new technique is presented to prove non-termination of term rewriting. The basic idea is to find a non-empty regular language of terms that is closed under rewriting and does not contain normal forms. It is automated by representing the language by a tree automaton with a fixed number of states, and expressing the mentioned requirements in a SAT formula. Satisfiability of this formula implies non-termination. Our approach succeeds for many examples where all earlier techniques fail, for instance for the *S*-rule from combinatory logic.

1998 ACM Subject Classification D.1.1 Applicative (Functional) Programming, D.3.1 Formal Definitions and Theory, F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems, I.1.1 Expressions and Their Representation, I.1.3 Languages and Systems

Keywords and phrases Non-termination, finite automata, regular languages

Digital Object Identifier 10.4230/LIPIcs.RTA.2015.160

## 1 Introduction

A basic approach for proving that a term rewriting system (TRS) is non-terminating is to prove that there is a regular language of terms that is closed under rewriting and does not contain normal forms. This is the main idea of the paper [26]. In this paper we present a new technique for proving non-termination of term rewriting systems (TRS) by using finite automata. The basic idea is to find a non-empty regular language of terms that is closed under rewriting and does not contain normal forms. It is automated by representing the language by a tree automaton with a fixed number of states, and expressing the mentioned requirements in a SAT formula. Satisfiability of this formula implies non-termination. Our approach succeeds for many examples where all earlier techniques fail, for instance for the *S*-rule from combinatory logic.



► w/ Jörg Endrullis:

## Non-Termination using Regular Languages

### Disproving Termination of Non-erasing Sole Combinatory Calculus with Tree Automata

Keisuke Nakano<sup>1</sup>✉ and Munehiro Iwami<sup>2</sup>

<sup>1</sup> Tohoku University, Sendai, Miyagi, Japan  
k.nakano@acm.org

<sup>2</sup> Shimane University, Matsue, Shimane, Japan  
munehiro@cis.shimane-u.ac.jp

**Abstract.** We study the termination of sole combinatory calculus, which consists of only one combinator. Specifically, the termination for non-erasing combinators is disproven by finding a desirable tree automaton with a SAT solver as done for term rewriting systems by Endrullis and Zantema. We improved their technique to apply to non-erasing sole combinatory calculus, in which it suffices to search for tree automata with a final sink state. Our method succeeds in disproving the termination of 8 combinators, whose termination has been an open problem.

**Keywords:** Combinatory calculus · Non-termination · Tree automata

#### 1 Introduction

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig



# IWC 2015

► Koji Nakazawa

Lambda Calculi and Confluence from A to Z



KOJI NAKAZAWA   
KEN-ETSU FUJITA

## Compositional Z: Confluence Proofs for Permutative Conversion

**Abstract.** This paper gives new confluence proofs for several lambda calculi with permutation-like reduction, including lambda calculi corresponding to intuitionistic and classical natural deduction with disjunction and permutative conversions, and a lambda calculus with explicit substitutions. For lambda calculi with permutative conversion, naive parallel reduction technique does not work, and (if we consider untyped terms, and hence we do not use strong normalization) traditional notion of residuals is required as Ando pointed out. This paper shows that the difficulties can be avoided by extending the technique proposed by Dehornoy and van Oostrom, called the Z theorem: existence of a mapping on terms with the Z property concludes the confluence. Since it is still hard to directly define a mapping with the Z property for the lambda calculi with permutative conversions, this paper extends the Z theorem to compositional functions, called compositional Z, and shows that we can adopt it to the calculi.

**Keywords:** Lambda calculus, Lambda-mu calculus, Confluence, Permutative conversion.

### 1. Introduction

The permutative conversion was introduced by Prawitz [12] as one of proof normalization techniques. The permutative conversion with disjunctions and ex-

# IWC 2015

- ▶ Koji Nakazawa  
Lambda Calculi and Confluence from A to Z
- ▶ Bertram Felgenhauer  
Point-Decreasing Diagrams Revisited

## ► Koji Nakazawa

Lambda Calculi and Confluence from A to Z

## ► Bertram Felgenhauer

Point-Decreasing Diagrams Revisited

Point-decreasing Diagrams Revisited

B. Felgenhauer

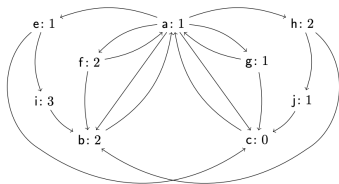


Figure 2: Labeling the “Maja the Bee”-example from [2].

Finally, we consider the question of completeness of the point version of decreasing diagrams.

**Theorem 5.** *Point-decreasing diagrams are complete for confluence of finite ARSs.*

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# IWC 2016

- ▶ Jörg Endrullis, Jan Willem Klop  
Roy Overbeek

Decreasing Diagrams: Two Labels Suffice

- Jörg Endrullis, Jan Willem Klop  
Roy Overbeek

## Decreasing Diagrams: Two Labels Suffice

### DECREASING DIAGRAMS FOR CONFLUENCE AND COMMUTATION

JÖRG ENDRULLIS<sup>a</sup>, JAN WILLEM KLOP<sup>a,b</sup>, AND ROY OVERBEEK<sup>a,b</sup>

<sup>a</sup> Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, the Netherlands

<sup>b</sup> Centrum Wiskunde & Informatica (CWI), Amsterdam, the Netherlands

*e-mail address:* j.endrullis@vu.nl

*e-mail address:* j.w.klop@vu.nl

*e-mail address:* r.overbeek@vu.nl

**ABSTRACT.** Like termination, confluence is a central property of rewrite systems. Unlike for termination, however, there exists no known complexity hierarchy for confluence. In this paper we investigate whether the decreasing diagrams technique can be used to obtain such a hierarchy. The decreasing diagrams technique is one of the strongest and most versatile methods for proving confluence of abstract rewrite systems. It is complete for countable systems, and it has many well-known confluence criteria as corollaries.

So what makes decreasing diagrams so powerful? In contrast to other confluence techniques, decreasing diagrams employ a labelling of the steps with labels from a well-founded order in order to conclude confluence of the underlying unlabelled relation. Hence it is natural to ask how the size of the label set influences the strength of the technique. In particular, what class of abstract rewrite systems can be proven confluent using decreasing diagrams restricted to 1 label, 2 labels, 3 labels, and so on? Surprisingly, we find that two labels suffice for proving confluence for every abstract rewrite system having the cofinality property, thus in particular for every confluent, countable system.

Secondly, we show that this result stands in sharp contrast to the situation for commutation of rewrite relations, where the hierarchy does not collapse.

Thirdly, investigating the possibility of a confluence hierarchy, we determine the first-order (non-)definability of the notion of confluence and related properties, using techniques from finite model theory. We find that in particular Hanf's theorem is fruitful for elegant

- ▶ Jörg Endrullis, Jan Willem Klop  
Roy Overbeek

Decreasing Diagrams: Two Labels Suffice

- ▶ Stefan Kahrs, Connor Smith

Non- $\omega$ -Overlapping TRSs are UN

## Non- $\omega$ -Overlapping TRSs are UN

Stefan Kahrs<sup>1</sup> and Connor Smith<sup>2</sup>

<sup>1</sup> School of Computing, University of Kent, Canterbury, United Kingdom  
S.M.Kahrs@kent.ac.uk

<sup>2</sup> School of Computing, University of Kent, Canterbury, United Kingdom  
cls204@kent.ac.uk

### Abstract

This paper solves problem #79 of RTA's list of open problems [14] – in the positive. If the rules of a TRS do not overlap w.r.t. substitutions of infinite terms then the TRS has unique normal forms. We solve the problem by reducing the problem to one of consistency for “similar” constructor term rewriting systems. For this we introduce a new proof technique. We define a relation  $\Downarrow$  that is consistent by construction, and which – if transitive – would coincide with the rewrite system's equivalence relation  $=_R$ .

We then prove the transitivity of  $\Downarrow$  by coalgebraic reasoning. Any concrete proof for instances of this relation only refers to terms of some finite coalgebra, and we then construct an equivalence relation on that coalgebra which coincides with  $\Downarrow$ .

**Keywords and phrases** consistency, omega-substitutions, uniqueness of normal forms

**Digital Object Identifier** 10.4230/LIPIcs.FSCD.2016.22

### 1 Introduction

For over 40 years [13] it has been known that TRSs that are left-linear and non-overlapping are confluent, and for over 30 years [8] that non-overlapping on its own may not even give us unique normal forms:



- ▶ Jörg Endrullis, Jan Willem Klop  
Roy Overbeek

Decreasing Diagrams: Two Labels Suffice

- ▶ Stefan Kahrs, Connor Smith

Non- $\omega$ -Overlapping TRSs are UN

- ▶ Thomas Sternagel, Christian Sternagel  
Formalized Confluence of Quasi-Reductive,  
Strongly Deterministic Conditional TRSs

## Certifying Confluence of Quasi-Decreasing Strongly Deterministic Conditional Term Rewrite Systems

Christian Sternagel<sup>(✉)</sup> and Thomas Sternagel<sup>(✉)</sup>

University of Innsbruck, Innsbruck, Austria  
{christian.sternagel,thomas.sternagel}@uibk.ac.at

**Abstract.** We formalize a confluence criterion for the class of quasi-decreasing strongly deterministic conditional term rewrite systems in Isabelle/HOL: confluence follows if all conditional critical pairs are joinable. However, quasi-decreasingness, strong determinism, and joinability of conditional critical pairs are all undecidable in general. Therefore, we also formalize sufficient criteria for those properties, which we incorporate into the general purpose certifier *CeTA* as well as the confluence checker *ConCon* for conditional term rewrite systems.

### 1 Introduction

In the area of equational reasoning *canonicity*—that is, termination together with confluence—plays an important role towards deciding equations with respect to equational theories and for avoiding redundant computations and nondeterminism. In the presence of powerful methods and tools for proving termination [1,1<sup>1</sup> 17 18,31,32], the remaining issue is to also establish

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# WST 2016

- ▶ w/ Alexander Fedotov:  
Non-Termination of String and Cycle  
Rewriting by Automata
- ▶ w/ Dennis Nolte, Barbara König:  
Termination of Term Graph Rewriting
- ▶ w/ David Sabel  
TermComp 2016 Participant: cycsrs 0.2

- ▶ w/ Alexander Fedotov:

## Non-Termination of String and Cycle Rewriting by Automata

- ▶ w/ Dennis Nolte, Barbara König:

## Termination of Term Graph Rewriting

- ▶ w/ David Sabel

## TermComp 2016 Participant: cycsrs 0.2

### TERMINATION OF CYCLE REWRITING BY TRANSFORMATION AND MATRIX INTERPRETATION

DAVID SABEL<sup>a</sup> AND HANS ZANTEMA<sup>b</sup>

<sup>a</sup> Goethe-University Frankfurt, Department of Computer Science and Mathematics, Computer Science Institute, 60629 Frankfurt am Main, Germany  
*e-mail address:* sabel@ki.informatik.uni-frankfurt.de

<sup>b</sup> TU Eindhoven, Department of Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
Radboud University Nijmegen, Institute for Computing and Information Sciences, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands  
*e-mail address:* h.zantema@tue.nl

**ABSTRACT.** We present techniques to prove termination of cycle rewriting, that is, string rewriting on cycles, which are strings in which the start and end are connected. Our main technique is to transform cycle rewriting into string rewriting and then apply state of the art techniques to prove termination of the string rewrite system. We present three such transformations, and prove for all of them that they are sound and complete. In this way not only termination of string rewriting of the transformed system implies termination of the original cycle rewrite system, a similar conclusion can be drawn for non-termination.

Apart from this transformational approach, we present a uniform framework of matrix interpretations, covering most of the earlier approaches to automatically proving termination of cycle rewriting.

All our techniques serve both for proving termination and relative termination.

We present several experiments showing the power of our techniques.

#### 1. INTRODUCTION

Cycles can be seen as strings of which the left end is connected to the right end, by which the string has no left end or right end any more. In Fig. 1 a pictorial representation of two such cycles is shown.

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# IWC 2017

- ▶ Julian Nagele, Aart Middeldorp  
CoCoWeb — A Convenient Web  
Interface for Confluence Tools

- ▶ Julian Nagele, Aart Middeldorp  
CoCoWeb — A Convenient Web  
Interface for Confluence Tools



## Cops and CoCoWeb: Infrastructure for Confluence Tools

Nao Hirokawa<sup>1</sup>✉, Julian Nagele<sup>2</sup>, and Aart Middeldorp<sup>3</sup>

<sup>1</sup> School of Information Science, JAIST, Nomi, Japan  
hirokawa@jaist.ac.jp

<sup>2</sup> School of Electronic Engineering and Computer Science,  
Queen Mary University of London, London, UK  
j.nagele@qmul.ac.uk

<sup>3</sup> Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
aart.middeldorp@uibk.ac.at

**Abstract.** In this paper we describe the infrastructure supporting confluence tools and competitions: Cops, the confluence problems database, and CoCoWeb, a convenient web interface for tools that participate in the annual confluence competition.

### 1 Introduction


In recent years several tools have been developed to automatically prove confluence and related properties of a variety of rewrite formats. These tools compete annually in the confluence competition [1] (CoCo).<sup>1</sup> Confluence tools run on confluence problems which are organized in the confluence problems (Cops) database. Cops is managed via a web interface

## ► Julian Nagele, Aart Middeldorp

### CoCoWeb — A Convenient Web Interface for Confluence Tools

CoCoWeb

COCO



[Home](#)
[Web Interface](#)

Tools

2023

2022

2021

2020

2019

CTRS

HRS

SRS

ACP

CSI

CoLL-Saigawa

noko-leipzig

TRS

2018

2017

2016

2015

2014

2013

2012

Enter a **rewrite system**, upload a file  or import a Cop:

```

1 (VAR x)
2 (RULES
3   a(b(x)) -> b(c(x))
4   c(b(x)) -> b(c(x))
5   c(b(x)) -> c(c(x))
6   b(b(x)) -> a(c(x))
7   a(b(x)) -> a(b(x))
8   c(c(x)) -> c(b(x))
9   a(c(x)) -> c(a(x))
10 )
11 (COMMENT
12   submitted by: Johannes Waldmann
13 )
14
```

property:

COM

CR

GCR

INF

NFP

UNC

UNR

timeout:

submit this problem to

Results

CR/2023/SRS/ACP

CR/2023/SRS/CONFident

CR/2023/SRS/CSI

CR/2019/SRS/noko-leipzig

Took 0.02s

NO

not joinable

CP

```

{ lhs = [b, c, b]
, rhs = [a, a, c]
, u1 = [a, b] -> [b, c]
, u2 = [b, b] -> [a, c]
, offset = 1}
```

universität  
innsbruck

3 September 2025

2. WST & IWC

IWC 2017 Oxford

AM  
66/100



- ▶ Julian Nagele, Aart Middeldorp

## CoCoWeb — A Convenient Web Interface for Confluence Tools

*Overigens ben ik wel onder de indruk hoe makkelijk je in cocoweb al de tools op dergelijke systemen online los kunt laten; in de terminatiewereld heb ik nooit iets gezien dat qua bruikbaarheid hierbij in de buurt kwam.*

*By the way, I am impressed how easily in cocoweb you can use tools on such systems online; in the termination world I've never seen anything that came close in terms of usability.*

11 September 2022

- ▶ Julian Nagele, Aart Middeldorp

## CoCoWeb — A Convenient Web Interface for Confluence Tools

*Overigens ben ik wel onder de indruk hoe makkelijk je in cocoweb al de tools op dergelijke systemen online los kunt laten; in de terminatiewereld heb ik nooit iets gezien dat qua bruikbaarheid hierbij in de buurt kwam.*

*By the way, I am impressed how easily in cocoweb you can use tools on such systems online; in the termination world I've never seen anything that came close in terms of usability.*

11 September 2022

`https://ari-web.uibk.ac.at/`

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# IWC 2018

- ▶ Christian Sternagel, Sarah Winkler  
Certified Ordered Completion

- ▶ Christian Sternagel, Sarah Winkler  
Certified Ordered Completion



## Certified Equational Reasoning via Ordered Completion

Christian Sternagel<sup>(✉)</sup> and Sarah Winkler<sup>(✉)</sup>

Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
{christian.sternagel,sarah.winkler}@uibk.ac.at

**Abstract.** On the one hand, equational reasoning is a fundamental part of automated theorem proving with ordered completion as a key technique. On the other hand, the complexity of corresponding, often highly optimized, automated reasoning tools makes implementations inherently error-prone. As a remedy, we provide a formally verified certifier for ordered completion based techniques. This certifier is code generated from an accompanying Isabelle/HOL formalization of ordered rewriting and ordered completion incorporating an advanced ground joinability criterion. It allows us to rigorously validate generated proof certificates from several domains: ordered completion, satisfiability in equational logic, and confluence of conditional term rewriting.

**Keywords:** Equational reasoning · Ordered completion · Ground joinability · Certification

### 1 Introduction

- ▶ Christian Sternagel, Sarah Winkler  
Certified Ordered Completion



## Certified Equational Reasoning via Ordered Completion

Christian Sternagel<sup>(✉)</sup><sup>1</sup> and Sarah Winkler<sup>(✉)</sup><sup>1</sup>

Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
{christian.sternagel,sarah.winkler}@uibk.ac.at

**Abstract.** On the one hand, equational reasoning is a fundamental part of automated theorem proving with ordered completion as a key technique. On the other hand, the complexity of corresponding, often highly optimized, automated reasoning tools makes implementations inherently error-prone. As a remedy, we provide a formally verified certifier for ordered completion based techniques. This certifier is code generated from an accompanying Isabelle/HOL formalization of ordered rewriting and ordered completion incorporating an advanced ground joinability criterion. It allows us to rigorously validate generated proof certificates from several domains: ordered completion, satisfiability in equational logic, and confluence of conditional term rewriting.

**Keywords:** Equational reasoning · Ordered completion · Ground joinability · Certification

### 1 Introduction

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Carsten Fuhs, Cynthia Kop

Improving Static Dependency Pairs for  
Higher-Order Dependency Pairs



- ▶ Carsten Fuhs, Cynthia Kop

## Improving Static Dependency Pairs for Higher-Order Dependency Pairs



### A Static Higher-Order Dependency Pair Framework

Carsten Fuhs<sup>1</sup>(✉) and Cynthia Kop<sup>2</sup>(✉)

<sup>1</sup> Department of Computer Science and Information Systems,  
Birkbeck, University of London, London, UK  
`carsten@dcsc.bbk.ac.uk`

<sup>2</sup> Department of Software Science, Radboud University Nijmegen,  
Nijmegen, The Netherlands  
`c.kop@cs.ru.nl`

**Abstract.** We revisit the static dependency pair method for proving termination of higher-order term rewriting and extend it in a number of ways: (1) We introduce a new rewrite formalism designed for general applicability in termination proving of higher-order rewriting, Algebraic Functional Systems with Meta-variables. (2) We provide a syntactically checkable soundness criterion to make the method applicable to a large class of rewrite systems. (3) We propose a modular dependency pair *framework* for this higher-order setting. (4) We introduce a fine-grained notion of *formative* and *computable* chains to render the framework more powerful. (5) We formulate several existing and new termination proving techniques in the form of processors within our framework.

The framework has been implemented in the (fully automatic) higher-order termination tool **WANDA**.

► Carsten Fuhs, Cynthia Kop

Improving Static Dependency Pairs for  
Higher-Order Dependency Pairs



# WST 2018

- ▶ Carsten Fuhs, Cynthia Kop  
Improving Static Dependency Pairs for  
Higher-Order Dependency Pairs
- ▶ Georg Moser, Michael Schaper  
TcT: Tyrolean Complexity Tool

- ▶ Carsten Fuhs, Cynthia Kop  
Improving Static Dependency Pairs for  
Higher-Order Dependency Pairs
- ▶ Georg Moser, Michael Schaper  
TcT: Tyrolean Complexity Tool

## TcT: Tyrolean Complexity Tool

Martin Avanzini<sup>1,2</sup>, Georg Moser<sup>3</sup>, and Michael Schaper<sup>3(✉)</sup>

<sup>1</sup> Università di Bologna, Bologna, Italy  
martin.avanzini@uibk.ac.at

<sup>2</sup> INRIA, Sophia Antipolis, France

<sup>3</sup> Department of Computer Science, University of Innsbruck, Innsbruck, Austria  
{georg.moser,michael.schaper}@uibk.ac.at

**Abstract.** In this paper we present  $\text{TcT}$  v3.0, the latest version of our fully automated complexity analyser.  $\text{TcT}$  implements our framework for automated complexity analysis and focuses on extensibility and automation.  $\text{TcT}$  is open with respect to the input problem under investigation and the resource metric in question. It is the most powerful tool in the realm of automated complexity analysis of term rewrite systems. Moreover it provides an expressive problem-independent strategy language that facilitates proof search. We give insights about design choices, the implementation of the framework and report different case studies where we have applied  $\text{TcT}$  successfully.

## 1 Introduction

Automatically checking programs for correctness has attracted the attention of the computer science research community since the birth of the discipline. Properties of interest are not necessarily functional, however, and among the non-functional ones, noticeable cases are bounds on the amount of resources (like time, memory, and program size) required to verify a property of verification.

- ▶ Carsten Fuhs, Cynthia Kop  
Improving Static Dependency Pairs for  
Higher-Order Dependency Pairs
- ▶ Georg Moser, Michael Schaper  
TcT: Tyrolean Complexity Tool



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Bertram Felgenhauer, Johannes Waldmann  
Proving Non-Joinability using Weakly  
Monotone Algebras

- ▶ Bertram Felgenhauer, Johannes Waldmann

## Proving Non-Joinability using Weakly Monotone Algebras

- ▶ Johannes Waldmann

## Noko-Leipzig at the 2019 Confluence Competition

### Noko-Leipzig at the 2019 Confluence Competition

Johannes Waldmann

Institut für Informatik, HTWK Leipzig, [johannes.waldmann@htwk-leipzig.de](mailto:johannes.waldmann@htwk-leipzig.de)

Noko-Leipzig is a confluence checker for string rewriting.

Noko-Leipzig implements a new method for proving non-joinability using arctically weighted automata [4], a generalisation of other methods using automata [1, 3]. In parallel, it checks local confluence and termination.

We found that even the basic method (no automata) is enough to answer 34 YES and 1401 NO for the 1541 string rewriting systems (SRS) from TPDB [2]. To get more interesting examples, we generated and filtered some random SRS, and submitted them for the Confluence Problems database. Among these are a few that can only be handled by the new method of weighted automata.

Noko-Leipzig uses the same code base as the Matchbox termination prover [5]. With competitor CSI [6], Noko-Leipzig shares the property that it rhymes with a TV series.

### References

- [1] T. Aoto. Disproving confluence of term rewriting systems by interpretation and ordering. In *Proc. 9th FroCoS*, volume 8152 of *LNCS (LNAI)*, pages 311–326, 2013.
- [2] A. Yamada et al. The termination problems data base. <http://www.termination-portal.org/wiki/TPDB>, 2019.
- [3] B. Felgenhauer and R. Thiemann. Reachability analysis with state-compatible automata. In *Proc. 8th LATA*, volume 8370 of *LNCS*, pages 347–359, 2013.
- [4] B. Felgenhauer and J. Waldmann. Proving non-joinability using weakly monotone algebras. Submitted, 2019.
- [5] J. Waldmann. The matchbox termination prover. <https://gitlab.imn.htwk-leipzig.de/waldmann/pure-matchbox>, 2019.
- [6] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A confluence tool. In *Proc. 23rd CADE*, volume 6803 of *LNCS (LNAI)*, pages 499–505, 2011.



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Kiraku Shintani, Nao Hirokawa  
Parallel Closedness Revisited

► Kiraku Shintani, Nao Hirokawa

## Parallel Closedness Revisited

### COMPOSITIONAL CONFLUENCE CRITERIA

KIRAKU SHINTANI ● AND NAO HIROKAWA ●

JAIST, Japan

*e-mail address:* s1820017@jaist.ac.jp, hirokawa@jaist.ac.jp

**ABSTRACT.** We show how confluence criteria based on decreasing diagrams are generalized to ones composable with other criteria. For demonstration of the method, the confluence criteria of orthogonality, rule labeling, and critical pair systems for term rewriting are recast into composable forms. We also show how such a criterion can be used for a reduction method that removes rewrite rules unnecessary for confluence analysis. In addition to them, we prove that Toyama's parallel closedness result based on parallel critical pairs subsumes his almost parallel closedness theorem.

#### 1. INTRODUCTION

Confluence is a property of rewriting that ensures uniqueness of computation results. In the last decades, various proof methods for confluence of term rewrite systems have been developed. They are roughly classified to three groups: (direct) confluence criteria based on critical pair analysis [KB70, Hue80, Toy81, Toy88, Gra96, vO97, Oku98, vO08, ZFM15], decomposition methods based on modularity and commutation [Toy87, AYT09, SH15], and transformation methods based on simulation of rewriting [AT12, Kah95, NFM15, SH15].

In this paper we present a confluence analysis based on *compositional* confluence criteria. Here a compositional criterion means a sufficient condition that, given a rewrite system  $R$ , its simulation  $C$  of  $R$  is confluent. Such a criterion is

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Carsten Fuhs

Between Derivational and Runtime  
Complexity

## Between Derivational and Runtime Complexity



### Transforming Derivational Complexity of Term Rewriting to Runtime Complexity

Carsten Fuhs<sup>(✉)</sup>

Department of Computer Science and Information Systems,  
Birkbeck, University of London, London, UK  
[carsten@dc.s.bbk.ac.uk](mailto:carsten@dc.s.bbk.ac.uk)

**Abstract.** Derivational complexity of term rewriting considers the length of the longest rewrite sequence for arbitrary start terms, whereas runtime complexity restricts start terms to basic terms. Recently, there has been notable progress in automatic inference of upper and lower bounds for runtime complexity. We propose a novel transformation that allows an off-the-shelf tool for inference of upper or lower bounds for runtime complexity to be used to determine upper or lower bounds for derivational complexity as well. Our approach is applicable to derivational complexity problems for innermost rewriting and for full rewriting. We have implemented the transformation in the tool APROVE and conducted an extensive experimental evaluation. Our results indicate that bounds for derivational complexity can now be inferred for rewrite systems that have been out of reach for automated analysis thus far.

## 1 Introduction

*Term rewrite systems (TRSs)* are a classic computational model both for equational reasoning and for computation. A TRS is a set of rewrite rules, which are used to transform terms into other terms. The complexity of a TRS is a measure of the resources required to compute the normal form of a term. Derivational complexity is the maximum length of a rewrite sequence starting from a given term. Runtime complexity is the maximum length of a rewrite sequence starting from a basic term. Basic terms are terms that do not contain any function symbols. The complexity of a TRS is a measure of the resources required to compute the normal form of a term. Derivational complexity is the maximum length of a rewrite sequence starting from a given term. Runtime complexity is the maximum length of a rewrite sequence starting from a basic term. Basic terms are terms that do not contain any function symbols.

- ▶ Carsten Fuhs

Between Derivational and Runtime  
Complexity

- ▶ Étienne Payet

Observing Loopingness

► Carsten Fuhs

Between Derivational and Runtime Complexity

► Étienne Payet

Observing Loopingness



## Non-termination in Term Rewriting and Logic Programming

Étienne Payet<sup>1</sup>

Received: 30 November 2021 / Accepted: 21 December 2023 / Published online: 2 February 2024  
 © The Author(s), under exclusive licence to Springer Nature B.V. 2024

### Abstract

In this paper, we define two particular forms of non-termination, namely *loops* and *binary chains*, in an abstract framework that encompasses term rewriting and logic programming. The definition of loops relies on the notion of *compatibility* of binary relations. We also present a syntactic criterion for the detection of a special case of binary chains. Moreover, we describe our implementation NTI and compare its results at the Termination Competition 2023 with those of leading analyzers.

**Keywords** Abstract reduction systems · Term rewriting · Logic programming · Non-termination · Loop

### 1 Introduction

This paper is concerned with the abstract treatment of non-termination in structures where one rewrites elements using indexed binary relations. Such structures can be formalised by *abstract reduction systems* (ARSs) [2, 30], i.e., pairs  $(A, \Rightarrow_\Pi)$  where  $A$  is a set and  $\Rightarrow_\Pi$  (the rewrite relation) is a union of binary relations on  $A$ , indexed by a set  $\Pi$ , i.e.,  $\Rightarrow_\Pi =$



- ▶ Carsten Fuhs  
Between Derivational and Runtime Complexity
- ▶ Étienne Payet  
Observing Loopingness
- ▶ Deivid Vale, Niels van der Weide  
Formalizing Higher-Order Termination in Coq

► Carsten Fuhs

## Between Derivational and Runtime Complexity


► Étienne Payet

## Observing Loopingness

► Deivid Vale, Niels van der Weide

## Formalizing Higher-Order Termination in Coq

## Certifying Higher-Order Polynomial Interpretations

Niels van der Weide ✉ 🏠 

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Deivid Vale ✉ 🏠 

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Cynthia Kop ✉ 🏠 

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

### Abstract

Higher-order rewriting is a framework in which one can write higher-order programs and study their properties. One such property is termination: the situation that for all inputs, the program eventually halts its execution and produces an output. Several tools have been developed to check whether higher-order rewriting systems are terminating. However, developing such tools is difficult and can be error-prone. In this paper, we present a way of certifying termination proofs of higher-order term rewriting systems. We formalize a specific method that is used to prove termination, namely the polynomial interpretation method. In addition, we give a program that processes proof traces containing a high-level description of a termination proof into a formal Coq proof script that can be checked by Coq. We demonstrate the usability of this approach by certifying higher-order polynomial interpretation proofs produced by Wanda, a termination analysis tool for higher-order rewriting.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Equational logic and rewriting

Keywords and phrases higher-order rewriting, Coq, termination, formalization

Digital Object Identifier 10.4230/LIPIcs.ITP.2023.30

Related Version Preprint version: <https://arxiv.org/abs/2302.11892> [32]

Supplementary Material Software (Coq Formalization): [github.com/nmvdw/Nijn](https://github.com/nmvdw/Nijn) [31]

Software (Proof script generator): [github.com/deividvale/nijn-coq-script-generation](https://github.com/deividvale/nijn-coq-script-generation) [30]

Funding Niels van der Weide: This research was supported by the NWO project “The Power of Equality” (NWO-M2). Deivid Vale: This research was supported by the Dutch Research Council (NWO).

► Carsten Fuhs

Between Derivational and Runtime Complexity

► Étienne Payet

Observing Loopingness

► Deivid Vale, Niels van der Weide

Formalizing Higher-Order Termination in Coq



## Certifying Higher-Order Polynomial Interpretations

Niels van der Weide ✉ 🏠 🇳🇱

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Deivid Vale ✉ 🏠 🇳🇱

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Cynthia Kop ✉ 🏠 🇳🇱

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

### Abstract

Higher-order rewriting is a framework in which one can write higher-order programs and study their properties. One such property is termination: the situation that for all inputs, the program eventually halts its execution and produces an output. Several tools have been developed to check whether higher-order rewriting systems are terminating. However, developing such tools is difficult and can be error-prone. In this paper, we present a way of certifying termination proofs of higher-order term rewriting systems. We formalize a specific method that is used to prove termination, namely the polynomial interpretation method. In addition, we give a program that processes proof traces containing a high-level description of a termination proof into a formal Coq proof script that can be checked by Coq. We demonstrate the usability of this approach by certifying higher-order polynomial interpretation proofs produced by Wanda, a termination analysis tool for higher-order rewriting.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Equational logic and rewriting

Keywords and phrases higher-order rewriting, Coq, termination, formalization

Digital Object Identifier 10.4230/LIPIcs.ITP.2023.30

Related Version Preprint version: <https://arxiv.org/abs/2302.11892> [32]

Supplementary Material Software (Coq Formalization): [github.com/nmvdw/Nijn](https://github.com/nmvdw/Nijn) [31]

Software (Proof script generator): [github.com/deividvale/nijn-coq-script-generation](https://github.com/deividvale/nijn-coq-script-generation) [30]

Funding Niels van der Weide: This research was supported by the NWO project “The Power of Equality” (NWO-M2) and Deivid Vale: This research was supported by the Dutch Research Council (NWO).

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# IWC 2021

- ▶ Alexander Lochmann, Fabian Mitterwallner, Aart Middeldorp  
CoCo 2021 Participant: FORTify 1.1

# IWC 2021

- ▶ Alexander Lochmann, Fabian Mitterwallner, Aart Middeldorp  
CoCo 2021 Participant: FORTify 1.1



# IWC 2021

- ▶ Alexander Lochmann, Fabian Mitterwallner, Aart Middeldorp  
CoCo 2021 Participant: FORTify 1.1



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig



- ▶ Christina Kohl, Aart Middeldorp

Development Closed Critical Pairs:  
Towards a Formalized Proof

► Christina Kohl, Aart Middeldorp

## Development Closed Critical Pairs: Towards a Formalized Proof

### A Formalization of the Development Closedness Criterion for Left-Linear Term Rewrite Systems

Christina Kohl

Department of Computer Science  
University of Innsbruck  
Innsbruck, Austria  
christina.kohl@uibk.ac.at

Aart Middeldorp

Department of Computer Science  
University of Innsbruck  
Innsbruck, Austria  
aart.middeldorp@uibk.ac.at

#### Abstract

Several critical pair criteria are known that guarantee confluence of left-linear term rewrite systems. The correctness of most of these have been formalized in a proof assistant. An important exception has been the development closedness criterion of van Oostrom. Its proof requires a high level of understanding about overlapping redexes and descendants as well as several intermediate results related to these concepts. We present a formalization in the proof assistant Isabelle/HOL. The result has been integrated into the certifier CeIA.

**CCS Concepts:** • Theory of computation → Equational logic and rewriting; Logic and verification.

**Keywords:** formalization, term rewriting, confluence

#### ACM Reference Format:

Christina Kohl and Aart Middeldorp. 2023. A Formalization of the Development Closedness Criterion for Left-Linear Term Rewrite Systems. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '23)*, January 16–17, 2023, Boston, MA, USA. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3573105.3575667>

#### 1 Introduction

Rewriting is a pervasive concept in mathematics, computer science, and other areas; simplification of expressions constitutes rewriting, the execution of a program can be seen as a rewrite sequence on program states, and in fact probably almost any development according to a set of fixed rules can be considered rewriting. In *term rewriting*, we assume that the objects which are rewritten are terms. This yields a powerful formalism which is crucial for the formalization of

verification, to name only a few application areas. In fact, term rewriting is a Turing-complete model of computation, and provides methods to investigate important properties of computation and simplification processes on an abstract level.

A term rewrite system  $\mathcal{R}$  is a set of directed equations, so-called rewrite rules, which induces a relation  $\rightarrow_{\mathcal{R}}$  on terms. Besides termination, which forbids infinite computations, confluence has been conceived as one of the central properties of rewriting. A rewrite system  $\mathcal{R}$  is confluent if for all terms  $s$ ,  $t$  and  $u$  such that  $s \rightarrow_{\mathcal{R}}^* t$  and  $s \rightarrow_{\mathcal{R}}^* u$  (here  $\rightarrow_{\mathcal{R}}^*$  denotes the transitive reflexive closure of  $\rightarrow_{\mathcal{R}}$ ) there exists a term  $v$  such that  $t \rightarrow_{\mathcal{R}}^* v$  and  $u \rightarrow_{\mathcal{R}}^* v$  (see Figure 2). Confluence is equivalent to the Church-Rosser property, introduced in 1936 by Church and Rosser [6] to show the consistency of the  $\lambda$ -calculus, and guarantees that normal forms (which are terms  $t$  such that  $t \rightarrow_{\mathcal{R}} u$  for no term  $u$ ) are unique.

Although undecidable in general, several sufficient conditions for confluence are known. The best-known ones are based on restricting the way in which critical pairs can be joined. These apply to left-linear rewrite systems and are covered in textbooks on term rewriting [5, 27]. The emergence of tools that aim to prove confluence automatically and compete in the Confluence Competition<sup>1</sup> (CoCo) [15] has led to many new techniques (e.g. [1, 3, 9, 11, 16]) also for rewrite systems that are not left-linear. Software tools may contain bugs and confluence tools are no exception. Indeed, YES/NO conflicts (i.e., instances where tools yield contradictory answers) are occasionally observed in CoCo, partly explaining the interest in certifying the output of confluence tools. An important first step is to formalize the techniques used in confluence tools in a proof assistant.

- Christina Kohl, Aart Middeldorp

## Development Closed Critical Pairs: Towards a Formalized Proof



## A Formalization of the Development Closedness Criterion for Left-Linear Term Rewrite Systems

Christina Kohl

Department of Computer Science  
University of Innsbruck  
Innsbruck, Austria  
christina.kohl@uibk.ac.at

Aart Middeldorp

Department of Computer Science  
University of Innsbruck  
Innsbruck, Austria  
aart.middeldorp@uibk.ac.at

### Abstract

Several critical pair criteria are known that guarantee confluence of left-linear term rewrite systems. The correctness of most of these have been formalized in a proof assistant. An important exception has been the development closedness criterion of van Oostrom. Its proof requires a high level of understanding about overlapping redexes and descendants as well as several intermediate results related to these concepts. We present a formalization in the proof assistant Isabelle/HOL. The result has been integrated into the certifier CeIA.

**CCS Concepts:** • Theory of computation → Equational logic and rewriting

### Key

### ACM Reference

Christina Kohl and Aart Middeldorp. Development Closedness Criterion for Left-Linear Term Rewrite Systems. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '23)*, January 16–17, 2023, Boston, MA, USA. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3573105.3575667>

### 1 Introduction

Rewriting is a pervasive concept in mathematics, computer science, and other areas; simplification of expressions constitutes rewriting, the execution of a program can be seen as a rewrite sequence on program states, and in fact probably almost any development according to a set of fixed rules can be considered rewriting. In *term rewriting*, we assume that the objects which are rewritten are terms. This yields a powerful formalism which is crucial for the formalization of

verification, to name only a few application areas. In fact, term rewriting is a Turing-complete model of computation, and provides methods to investigate important properties of computation and simplification processes on an abstract level.

A term rewrite system  $\mathcal{R}$  is a set of directed equations, so-called rewrite rules, which induces a relation  $\rightarrow_{\mathcal{R}}$  on terms. Besides termination, which forbids infinite computations, confluence has been conceived as one of the central properties of rewriting. A rewrite system  $\mathcal{R}$  is confluent if for all terms  $s, t$  and  $u$  such that  $s \rightarrow_{\mathcal{R}}^* t$  and  $s \rightarrow_{\mathcal{R}}^* u$  (here  $\rightarrow_{\mathcal{R}}^*$  denotes the transitive reflexive closure of  $\rightarrow_{\mathcal{R}}$ ) there exists a term  $v$  such that  $t \rightarrow_{\mathcal{R}}^* v$  and  $u \rightarrow_{\mathcal{R}}^* v$  (see Figure 2). Confluence

distinguished paper award

is a property introduced in the theory of rewriting, which is unique. In general, several sufficient conditions for confluence are known. The best-known ones are based on restricting the way in which critical pairs can be joined. These apply to left-linear rewrite systems and are covered in textbooks on term rewriting [5, 27]. The emergence of tools that aim to prove confluence automatically and compete in the Confluence Competition<sup>1</sup> (CoCo) [15] has led to many new techniques (e.g. [1, 3, 9, 11, 16]) also for rewrite systems that are not left-linear. Software tools may contain bugs and confluence tools are no exception. Indeed, YES/NO conflicts (i.e., instances where tools yield contradictory answers) are occasionally observed in CoCo, partly explaining the interest in certifying the output of confluence tools. An important first step is to formalize the techniques used in confluence tools in a proof assistant.

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

► Nao Hirokawa, Aart Middeldorp

## Hydra Battles and AC Termination

Logical Methods in Computer Science  
Volume 21, Issue 2, 2025, pp. 29:1–29:22  
<https://lmcs.episciences.org/>

Submitted Mar. 06, 2024  
Published Jun. 27, 2025

### HYDRA BATTLES AND AC TERMINATION

NAO HIROKAWA <sup>a</sup> AND AART MIDDELDORP  <sup>b</sup>

<sup>a</sup> School of Information Science, JAIST, Japan  
*e-mail address:* hirokawa@jaist.ac.jp

<sup>b</sup> Department of Computer Science, University of Innsbruck, Austria  
*e-mail address:* aart.middeldorp@uibk.ac.at

**ABSTRACT.** We present a new encoding of the Battle of Hercules and Hydra as a rewrite system with AC symbols. Unlike earlier term rewriting encodings, it faithfully models any strategy of Hercules to beat Hydra. To prove the termination of our encoding, we employ type introduction in connection with many-sorted semantic labeling for AC rewriting and AC-MPO, a new AC compatible reduction order that can be seen as a much weakened version of AC-RPO.

#### 1. INTRODUCTION

The mythological monster Hydra is a dragon-like creature with multiple heads. Whenever Hercules in his fight chops off a head, more and more new heads can grow instead, since the beast gets increasingly angry. Here we model a Hydra as an unordered tree. If Hercules cuts off a leaf corresponding to a head, the tree is modified in the following way: If the cut-off node  $h$  has a grandparent  $n$ , then the branch from  $n$  to the parent of  $h$  gets multiplied, where the number of copies corresponds to the number of decapitations so far. Hydra dies if there are no heads left, in that case Hercules wins. The following sequence shows an example fight:



- ▶ Nao Hirokawa, Aart Middeldorp  
Hydra Battles and AC Termination
- ▶ challenge for termination tools

$$\begin{aligned}A(n, i(h)) &\rightarrow A(s(n), h) \\A(n, i(h \mid x)) &\rightarrow A(s(n), i(x)) \\A(n, i(x)) &\rightarrow B(n, D(s(n), i(x))) \\C(0, x) &\rightarrow E(x) \\C(s(n), x) &\rightarrow x \mid C(n, x) \\i(E(x) \mid y) &\rightarrow E(i(x \mid y)) \\i(E(x)) &\rightarrow E(i(x)) \quad \text{AC}(\mid) \\D(n, i(i(x))) &\rightarrow i(D(n, i(x))) \\D(n, i(i(x) \mid y)) &\rightarrow i(D(n, i(x)) \mid y) \\D(n, i(i(h \mid x) \mid y)) &\rightarrow i(C(n, i(x)) \mid y) \\D(n, i(i(h \mid x))) &\rightarrow i(C(n, i(x))) \\D(n, i(i(h) \mid y)) &\rightarrow i(C(n, h) \mid y) \\D(n, i(i(h))) &\rightarrow i(C(n, h)) \\B(n, E(x)) &\rightarrow A(s(n), x)\end{aligned}$$

- ▶ Nao Hirokawa, Aart Middeldorp  
Hydra Battles and AC Termination

AAECC 7, 133–162 (1996)



## Total Termination of Term Rewriting

M.C.F. Ferreira\*, H. Zantema

Utrecht University, Department of Computer Science, P.O. Box 80.089, 3509 TB Utrecht, The Netherlands, {maria, hanzs}@cs.ruu.nl

Received December 7, 1993; revised version January 3, 1995

**Abstract.** We investigate proving termination of term rewriting systems by a compositional interpretation of terms in a total well-founded order. This kind of termination is called *total termination*. Equivalently total termination can be characterized by the existence of an order on ground terms which is total, well-founded and closed under contexts. For finite signatures, total termination implies simple termination. The converse does not hold. However, total termination generalizes most of the usual techniques for proving termination (including the well-known recursive path order). It turns out that for this kind of termination the only interesting orders below  $\varepsilon_0$  are built from the natural numbers by lexicographic product and the multiset construction. By examples we show that both constructions are essential. Most of the techniques used are based on ordinal arithmetic.

## ► Nao Hirokawa, Aart Middeldorp

### Hydra Battles and AC Termination

We end this section with an example based on the battle of Hercules and the Hydra (see [11]; another version of this game appears in [3]). For this system we conjecture  $u_R = \varepsilon_0$ .

The Hydra is represented as a finite tree. We code the tree using a binary symbol  $c$ : a tree consisting of a root and descendants  $t_1, \dots, t_k$  is represented as  $c(t_1, c(t_2, \dots, c(t_{k-1}, t_k) \dots))$ , that is  $c(D, S)$  represents a node whose descendants are coded in the subtree  $D$  and whose siblings are coded in subtree  $S$ . Leaves are represented by the constant  $nil$ .

On each stage, a leaf node is selected and deleted. Afterwards,  $k \geq 0$  copies of the subtree containing the now missing leaf, are added to the second ancestor of the selected leaf. The number of copies is chosen randomly. The game can be represented as the infinite TRS  $H$ :

$$\begin{aligned}
 c(nil, x) &\rightarrow x \\
 c(c(nil, x), y) &\rightarrow copy(n, x, y) \\
 copy(s(k), x, y) &\rightarrow copy(k, x, c(x, y)) \\
 copy(0, x, y) &\rightarrow y \\
 n &\rightarrow s^i(0) \quad \text{for each } i \geq 0
 \end{aligned}$$

## Total Termination of Term Rewriting

M.C.F. Ferreira\*, H. Zantema

Utrecht University, Department of Computer Science, P.O. Box 80.089, 3509 TB Utrecht, The Netherlands, {maria, hanzs}@cs.ruu.nl

Received December 7, 1993; revised version January 3, 1995

**Abstract.** We investigate proving termination of term rewriting systems by a compositional interpretation of terms in a total well-founded order. This kind of termination is called *total termination*. Equivalently total termination can be characterized by the existence of an order on ground terms which is total, well-founded and closed under contexts. For finite signatures, total termination implies simple termination. The converse does not hold. However, total termination generalizes most of the usual techniques for proving termination (including the well-known recursive path order). It turns out that for this kind of termination the only interesting orders below  $\varepsilon_0$  are built from the natural numbers by lexicographic product and the multiset construction. By examples we show that both constructions are essential. Most of the techniques used are based on ordinal arithmetic.



► Nao Hirokawa, Aart Middeldorp

## Hydra Battles and AC Termination

We end this section with an example based on the battle of Hercules and the Hydra (see [11]; another version of this game appears in [3]). For this system we conjecture  $u_R = \varepsilon_0$ .

The Hydra is represented as a finite tree. We code the tree using a binary symbol  $c$ : a tree consisting of a root and descendants  $t_1, \dots, t_k$  is represented as  $c(t_1, c(t_2, \dots, c(t_{k-1}, t_k) \dots))$ , that is  $c(D, S)$  represents a node whose descendants are coded in the subtree  $D$  and whose siblings are coded in subtree  $S$ . Leaves are represented by the constant  $nil$ .

On each stage, a leaf node is selected and deleted. Afterwards,  $k \geq 0$  copies of the subtree containing the now missing leaf, are added to the second ancestor of the selected leaf. The number of copies is chosen randomly. The game can be represented as the infinite TRS  $H$ :

$$\begin{aligned} c(nil, x) &\rightarrow x \\ c(c(nil, x), y) &\rightarrow copy(n, x, y) \\ copy(s(k), x, y) &\rightarrow copy(k, x, c(x, y)) \\ copy(0, x, y) &\rightarrow y \\ n &\rightarrow s^i(0) \quad \text{for each } i \geq 0 \end{aligned}$$

AAECC 7, 133–162 (1996)



## Total Termination of Term Rewriting

M.C.F. Ferreira\*, H. Zantema

Utrecht University, Department of Computer Science, P.O. Box 80.089, 3509 TB Utrecht, The Netherlands, {maria, hanzs}@cs.ruu.nl

Received December 7, 1993; revised version January 3, 1995

**Abstract.** We investigate proving termination of term rewriting systems by a compositional interpretation of terms in a total well-founded order. This kind of termination is called *total termination*. Equivalently total termination can be characterized by the existence of an order on ground terms which is total, well-founded and closed under contexts. For finite signatures, total termination implies simple termination. The converse does not hold. However, total termination generalizes most of the usual techniques for proving termination (including the well-known recursive path order). It turns out that for this kind of termination the only interesting orders below  $\varepsilon_0$  are built from the natural numbers by lexicographic product and the multiset construction. By examples we show that both constructions are essential. Most of the techniques used are based on ordinal arithmetic.

- ▶ Nao Hirokawa, Aart Middeldorp
- Hydra Battles and AC Termination



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Takahito Aoto et al  
A New Format for Rewrite Systems



► Takahito Aoto et al

## A New Format for Rewrite Systems



	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

- ▶ Fabian Mitterwallner, Aart Middeldorp  
René Thiemann  
Linear Termination over  $\mathbb{N}$  is Undecidable

► Fabian Mitterwallner, Aart Middeldorp  
René Thiemann

## Linear Termination over $\mathbb{N}$ is Undecidable

### Linear Termination is Undecidable

Fabian Mitterwallner

Aart Middeldorp

René Thiemann

fabian.mitterwallner@uibk.ac.at

aart.middeldorp@uibk.ac.at

rene.thiemann@uibk.ac.at

University of Innsbruck, Department of Computer Science  
Innsbruck, Austria

#### ABSTRACT

By means of a simple reduction from Hilbert's 10th problem we prove the somewhat surprising result that termination of *one-rule* rewrite systems by a *linear* interpretation in the natural numbers is undecidable. The very same reduction also shows the undecidability of termination of one-rule rewrite systems using the Knuth-Bendix order with *subterm coefficients*. The linear termination problem remains undecidable for one-rule rewrite systems that can be shown terminating by a (non-linear) polynomial interpretation. We further show the undecidability of the problem whether a one-rule rewrite system can be shown terminating by a polynomial interpretation with *rational* or *real* coefficients. Several of our results have been formally verified in the Isabelle/HOL proof assistant.

#### CCS CONCEPTS

• Theory of computation → Equational logic and rewriting; Computability.

#### KEYWORDS

term rewriting, polynomial termination, undecidability

#### ACM Reference Format:

Fabian Mitterwallner, Aart Middeldorp, and René Thiemann. 2024. Linear Termination is Undecidable. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24)*, July 8–11, 2024, Tallinn, Estonia. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3661814.3662081>

#### 1 INTRODUCTION

In this paper we consider the (uniform) termination problem for first-order term rewrite systems. Using a reduction from the halting problem for Turing machines, Huët and Lankford [18] were the first to show the undecidability of the termination problem. Dauchet [9] proved that the termination problem remains undecidable for one-

Decades of research have been devoted to develop powerful sufficient conditions that are amenable to automation. In this paper we are concerned with one of the earliest termination methods: using a polynomial interpretation over the natural numbers, which goes back to Lankford [23]. Two problems need to be addressed when using polynomial interpretations for proving termination, whether by hand or by a tool: (1) finding suitable polynomials for the function symbols, and (2) showing that the induced order constraints on polynomials are valid. Heuristics for the former problem are presented in [7, 41]. The latter problem amounts to proving  $P(x_1, \dots, x_n) > 0$  for all natural numbers  $x_1, \dots, x_n \in \mathbb{N}$ , for polynomials  $P \in \mathbb{Z}[x_1, \dots, x_n]$ . This is known to be undecidable, as an easy consequence of the undecidability of Hilbert's 10th problem, see e.g., Zantema [41, Proposition 6.2.11]. In a recent paper [31] the first two authors proved by a non-trivial reduction from Hilbert's 10th problem that (1) is undecidable. In this paper we prove the somewhat surprising result that (1) remains undecidable if we restrict the allowed interpretation functions to linear ones, even for one-rule rewrite systems that can be shown terminating by a polynomial interpretation. This contrasts with the fact that (2) is decidable for linear interpretations. We further show that the problem is undecidable when strict monotonicity of the interpretation functions is weakened to weak monotonicity. This is relevant for automated termination proving, e.g. when dependency pairs [1] are used.

Polynomial termination over the natural numbers is the strongest property in the following hierarchy of termination [41]

$$PT \Rightarrow \omega T \Rightarrow TT \Rightarrow ST \Rightarrow SN \quad (*)$$

where the acronyms from left to right stand for the following properties: Polynomial termination over  $\mathbb{N}$  (PT),  $\omega$ -termination ( $\omega T$ ), total termination (TT), simple termination (ST) and termination (SN). All properties in this hierarchy are known to be undecidable.



► Fabian Mitterwallner, Aart Middeldorp  
René Thiemann

## Linear Termination over $\mathbb{N}$ is Undecidable

### Linear Termination is Undecidable

Fabian Mitterwallner

Aart Middeldorp

René Thiemann

fabian.mitterwallner@uibk.ac.at

aart.middeldorp@uibk.ac.at

rene.thiemann@uibk.ac.at

University of Innsbruck, Department of Computer Science  
Innsbruck, Austria

#### ABSTRACT

By means of a simple reduction from Hilbert's 10th problem we prove the somewhat surprising result that termination of *one-rule* rewrite systems by a *linear* interpretation in the natural numbers is undecidable. The very same reduction also shows the undecidability of termination of one-rule rewrite systems using the Knuth-Bendix order with *subterm coefficients*. The linear termination problem remains undecidable for one-rule rewrite systems that can be shown terminating by a (non-linear) polynomial interpretation. We further show the undecidability of the problem whether a one-rule rewrite system can be shown terminating by a polynomial interpretation with *rational* or *real* coefficients. Several of our results have been formally verified in the Isabelle/HOL proof assistant.

#### CCS CONCEPTS

• Theory of computation → Equational logic and rewriting; Computability.

#### KEYWORDS

term rewriting, polynomial termination, undecidability

#### ACM Reference Format:

Fabian Mitterwallner, Aart Middeldorp, and René Thiemann. 2024. Linear Termination is Undecidable. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24)*, July 8–11, 2024, Tallinn, Estonia. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3661814.3662081>

#### 1 INTRODUCTION

In this paper we consider the (uniform) termination problem for first-order term rewrite systems. Using a reduction from the halting problem for Turing machines, Huët and Lankford [18] were the first to show the undecidability of the termination problem. Dauchet [9] proved that the termination problem remains undecidable for one-

Decades of research have been devoted to develop powerful sufficient conditions that are amenable to automation. In this paper we are concerned with one of the earliest termination methods: using a polynomial interpretation over the natural numbers, which goes back to Lankford [23]. Two problems need to be addressed when using polynomial interpretations for proving termination, whether by hand or by a tool: (1) finding suitable polynomials for the function symbols, and (2) showing that the induced order constraints on polynomials are valid. Heuristics for the former problem are presented in [7, 41]. The latter problem amounts to proving  $P(x_1, \dots, x_n) > 0$  for all natural numbers  $x_1, \dots, x_n \in \mathbb{N}$ , for polynomials  $P \in \mathbb{Z}[x_1, \dots, x_n]$ . This is known to be undecidable, as an easy consequence of the undecidability of Hilbert's 10th problem, see e.g., Zantema [41, Proposition 6.2.11]. In a recent paper [31] the first two authors proved by a non-trivial reduction from Hilbert's 10th problem that (1) is undecidable. In this paper we prove the somewhat surprising result that (1) remains undecidable if we restrict the allowed interpretation functions to linear ones, even for one-rule rewrite systems that can be shown terminating by a polynomial interpretation. This contrasts with the fact that (2) is decidable for linear interpretations. We further show that the problem is undecidable when strict monotonicity of the interpretation functions is weakened to weak monotonicity. This is relevant for automated termination proving, e.g. when dependency pairs [1] are used.

Polynomial termination over the natural numbers is the strongest property in the following hierarchy of termination [41]

$$\text{PT} \Rightarrow \omega\text{T} \Rightarrow \text{TT} \Rightarrow \text{ST} \Rightarrow \text{SN} \quad (*)$$

where the acronyms from left to right stand for the following properties: Polynomial termination over  $\mathbb{N}$  (PT),  $\omega$ -termination ( $\omega$ T), total termination (TT), simple termination (ST) and termination (SN). All properties in this hierarchy are known to be undecidable.

- ▶ Fabian Mitterwallner, Aart Middeldorp  
René Thiemann

Linear Termination over  $\mathbb{N}$  is Undecidable

LT  $\Rightarrow$  PT  $\Rightarrow$   $\omega$ T  $\Rightarrow$  TT  $\Rightarrow$  ST  $\Rightarrow$  SN

- ▶ Fabian Mitterwallner, Aart Middeldorp  
René Thiemann  
Linear Termination over  $\mathbb{N}$  is Undecidable
- ▶ Dieter Hofbauer  
MultumNonMultum entering Term Rewriting

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

► levgen Ivanov

On Non-Triviality of the Hierarchy of  
Decreasing Church–Rosser Abstract  
Rewriting Systems

► Ievgen Ivanov

## On Non-Triviality of the Hierarchy of Decreasing Church–Rosser Abstract Rewriting Systems

### Completeness of the Decreasing Diagrams Method for Proving Confluence of Rewriting Systems of the Least Uncountable Cardinality

Ievgen Ivanov ✉

Taras Shevchenko National University of Kyiv, Ukraine

#### Abstract

We show that every confluent abstract rewriting system (ARS) of the cardinality that does not exceed the first uncountable cardinal belongs to the class  $DCR_3$ , i.e. the class of confluent ARS for which confluence can be proved with the help of the decreasing diagrams method using the set of labels  $\{0,1,2\}$  ordered in such a way that  $0 < 1 < 2$  (in the general case, the decreasing diagrams method with two labels is not sufficient for proving confluence of such ARS). Under the Continuum Hypothesis this result implies that the decreasing diagrams method is sufficient for establishing confluence of ARS on many structures of interest to applied mathematics and various interdisciplinary fields (confluence of ARS on real numbers, continuous real functions, etc.).

We provide a machine-checked formal proof of a formalized version of the main result in Isabelle proof assistant using HOL logic and the HOL-Cardinals theory. An extended version of this formalization is available in the Archive of Formal Proofs.

**2012 ACM Subject Classification** Theory of computation → Equational logic and rewriting; Theory of computation → Logic and verification

**Keywords and phrases** confluence, decreasing diagrams method, rewriting systems, reduction, formal methods, formal proofs, formal verification, non-discrete models, nondeterministic models, interval models

**Digital Object Identifier** 10.4230/LIPICs.FSCD.2025.25

**Related Version** *Extended Formalization:*

[https://isa-afp.org/entries/Completeness\\_Decreasing\\_Diagrams\\_for\\_N1.html](https://isa-afp.org/entries/Completeness_Decreasing_Diagrams_for_N1.html)

**Supplementary Material** *Text:* <https://doi.org/10.5281/zenodo.14254256> [17]

	WST	IWC
1993	St. Andrews	
1995	La Bresse	
1997	Ede	
1999	Dagstuhl	
2001	Utrecht	
2003	Valencia	
2004	Aachen	
2006	Seattle	
2007	Paris	
2009	Leipzig	
2010	Edinburgh	
2012	Obergurgl	Nagoya

	WST	IWC
2013	Bertinoro	Eindhoven
2014	Vienna	Vienna
2015		Berlin
2016	Obergurgl	Obergurgl
2017		Oxford
2018	Oxford	Oxford
2019		Dortmund
2020		Paris
2021	Pittsburgh	Buenos Aires
2022	Haifa	Haifa
2023	Obergurgl	Obergurgl
2024		Tallinn
2025	Leipzig	Leipzig

# Outline

1. Hans Zantema
2. WST & IWC
- 3. Hans Zantema**
4. Workshops and Competitions
5. Hans Zantema



## Termination of Context-Sensitive Rewriting

H. Zantema

Utrecht University, Department of Computer Science,  
P.O. box 80.089, 3508 TB Utrecht, The Netherlands  
e-mail: [hansz@cs.ruu.nl](mailto:hansz@cs.ruu.nl)

### Abstract

Context-sensitive term rewriting is a kind of term rewriting in which reduction is not allowed inside some fixed arguments of some function symbols. We introduce two new techniques for proving termination of context-sensitive rewriting. The first one is a modification of the technique of interpretation in a well-founded order, the second one is implied by a transformation in which context-sensitive termination of the original system can be concluded from termination of the transformed one. In combination with purely automatic techniques for proving ordinary termination, the latter technique is purely automatic too.

## 1 Introduction

The function computing the factorial is usually defined as follows:

$$\text{fact}(x) = \text{if } x = 0, 1 \text{ then } x * \text{fact}(x - 1),$$

## Generalized Innermost Rewriting

Jaco van de Pol<sup>1,2</sup> and Hans Zantema<sup>2</sup>

<sup>1</sup> Department of Software Engineering, CWI, P.O. Box 94.079,  
1090 GB Amsterdam, The Netherlands  
[Jaco.van.de.Pol@cwi.nl](mailto:Jaco.van.de.Pol@cwi.nl)

<sup>2</sup> Department of Computer Science, TU Eindhoven, P.O. Box 513,  
5600 MB Eindhoven, The Netherlands  
[H.Zantema@tue.nl](mailto:H.Zantema@tue.nl)

**Abstract.** We propose two generalizations of innermost rewriting for which we prove that termination of innermost rewriting is equivalent to termination of generalized innermost rewriting. As a consequence, by rewriting in an arbitrary TRS certain non-innermost steps may be allowed by which the termination behavior and efficiency is often much better, but never worse than by only doing innermost rewriting.

## 1 Introduction

In term rewriting one can rewrite according various reduction strategies, for instance innermost or outermost. It may occur that by one strategy a normal form is reached while rewriting according another strategy may go on forever. For instance, innermost rewriting of  $f(a)$  by the TRS consisting of the two rules  $a \rightarrow b, f(a) \rightarrow f(a)$  yields a normal form in one step while outermost rewriting goes on forever. By the TRS consisting of the two rules  $a \rightarrow a, f(x) \rightarrow b$  the behavior of  $f(a)$  is opposite: now innermost rewriting goes on forever while outermost rewriting yields a normal form in one step. We say that a particular strategy is not worse than another strategy if for every term for which the latter yields a normal form, the same holds for the former. The above examples

# Triangulation in Rewriting

Vincent van Oostrom<sup>1</sup> and Hans Zantema<sup>2</sup>

- 1 Department of Philosophy, Utrecht University, The Netherlands  
Vincent.vanOostrom@phil.uu.nl
- 2 Department of Computer Science, TU Eindhoven, The Netherlands  
Institute for Computing and Information Sciences, Radboud University  
Nijmegen, The Netherlands  
h.zantema@tue.nl

## Abstract

We introduce a process, dubbed *triangulation*, turning any rewrite relation into a confluent one. It is more direct than usual completion, in the sense that objects connected by a peak are directly related rather than their normal forms. We investigate conditions under which this process preserves desirable properties such as termination.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases triangulation, codeterminism, completion, (co)confluence, (co)termination

Digital Object Identifier 10.4230/LIPIcs.RTA.2012.240

Category Regular Research Paper

## 1 Introduction

We study the problem of deciding whether two objects are equivalent with respect to the equivalence relation generated by a rewrite relation. We do this in a fully abstract

Applicable Algebra in Engineering, Communication and Computing (2025) 36:341–363  
https://doi.org/10.1007/s00200-023-00606-6

## ORIGINAL PAPER



# The paint pot problem and common multiples in monoids

Hans Zantema<sup>1,2</sup> · Vincent van Oostrom<sup>3</sup>

Received: 28 July 2022 / Accepted: 3 April 2023 / Published online: 13 May 2023  
© The Author(s) 2023, corrected publication 2023

## Abstract

Illustrated by a problem on paint pots that is easy to understand but hard to solve, we investigate whether particular monoids have the property of common right multiples. As one result we characterize generalized braid monoids represented by undirected graphs, being a subclass of Artin–Tits monoids. Stated in other words, we investigate to which graphs the old Garside result stating that braid monoids have the property of common right multiples, generalizes. This characterization also follows from old results on Coxeter groups and the connection between finiteness of Coxeter groups and common right multiples in Artin–Tits monoids. However, our independent presentation is self-contained up to some basic knowledge of rewriting, and also applies to monoids beyond the Artin–Tits format. The main new contribution is a technique to prove that the property of common right multiples does not hold, by finding a particular model, in our examples all being finite.

**Keywords** Artin–Tits monoids · Generalized braids · Rewriting · Common multiples · Tiling

# Outline

1. Hans Zantema
2. WST & IWC
3. Hans Zantema
- 4. Workshops and Competitions**
5. Hans Zantema

## Workshops (Selection)

► HOR

► IWC

► WPTE

► WST

## Workshops (Selection)

- ▶ HOR
- ▶ **IWC**
- ▶ WPTE
- ▶ WST

## International Workshop on Confluence

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <http://cl-informatik.uibk.ac.at/iwc/>
- ▶ friendly workshop with clear bylaws and steering committee

## Workshops (Selection)

- ▶ HOR
- ▶ IWC
- ▶ WPTE
- ▶ WST

## International Workshop on Confluence

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <http://cl-informatik.uibk.ac.at/iwc/>
- ▶ friendly workshop with clear **bylaws** and steering committee

## Workshops (Selection)

- ▶ HOR
- ▶ **IWC**
- ▶ WPTE
- ▶ WST

## International Workshop on Confluence

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <http://cl-informatik.uibk.ac.at/iwc/>
- ▶ friendly workshop with clear **bylaws** and steering committee

## Bylaws

- 1 The International Confluence Workshop (IWC) is an annual workshop.

## Workshops (Selection)

- ▶ HOR
- ▶ **IWC**
- ▶ WPTE
- ▶ WST

## International Workshop on Confluence

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <http://cl-informatik.uibk.ac.at/iwc/>
- ▶ friendly workshop with clear **bylaws** and steering committee

## Bylaws

- 1 The International Confluence Workshop (IWC) is an annual workshop.
- 2 IWC has a steering committee (SC) consisting of two members. Each SC member serves for five consecutive workshops.



## Workshops (Selection)

- ▶ HOR
- ▶ IWC
- ▶ WPTE
- ▶ WST

## International Workshop on Confluence

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <http://cl-informatik.uibk.ac.at/iwc/>
- ▶ friendly workshop with clear **bylaws** and steering committee

## Bylaws

- 1 The International Confluence Workshop (IWC) is an annual workshop.
- 2 IWC has a steering committee (SC) consisting of two members. Each SC member serves for five consecutive workshops.
- 3 Each IWC has two program committee (PC) chairs. Each PC chair serves two consecutive workshops.

- 4 PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.

## Bylaws (cont'd)

- ④ PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.
- ⑤ IWC is an informal workshop without formal (post)proceedings.

## Bylaws (cont'd)

- ④ PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.
- ⑤ IWC is an informal workshop without formal (post)proceedings.
- ⑥ Reviewing of submitted papers is done by the PC. Reviews should be fair and constructive. An effort should be made to accommodate all submitted papers that are not incorrect and in scope of the workshop.

## Bylaws (cont'd)

- ④ PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.
- ⑤ IWC is an informal workshop without formal (post)proceedings.
- ⑥ Reviewing of submitted papers is done by the PC. Reviews should be fair and constructive. An effort should be made to accommodate all submitted papers that are not incorrect and in scope of the workshop.
- ⑦ The SC decides the location of the IWC to take place in the next year and appoints one new PC chair to replace the PC chair that served two terms. SC members cannot be PC chair.

- ④ PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.
- ⑤ IWC is an informal workshop without formal (post)proceedings.
- ⑥ Reviewing of submitted papers is done by the PC. Reviews should be fair and constructive. An effort should be made to accommodate all submitted papers that are not incorrect and in scope of the workshop.
- ⑦ The SC decides the location of the IWC to take place in the next year and appoints one new PC chair to replace the PC chair that served two terms. SC members cannot be PC chair.
- ⑧ The SC assists the PC chairs with tasks like workshop proposal writing (when IWC is to collocate with a conference) and coordination with the confluence competition (CoCo). The SC is responsible for archiving the (informal) workshop proceedings.

## Bylaws (cont'd)

- 4 PC chairs are responsible for the program (invited speakers, web site, publicity) and carry the sole financial responsibility. The PC chairs select the other PC members.
- 5 IWC is an informal workshop without formal (post)proceedings.
- 6 Reviewing of submitted papers is done by the PC. Reviews should be fair and constructive. An effort should be made to accommodate all submitted papers that are not incorrect and in scope of the workshop.
- 7 The SC decides the location of the IWC to take place in the next year and appoints one new PC chair to replace the PC chair that served two terms. SC members cannot be PC chair.
- 8 The SC assists the PC chairs with tasks like workshop proposal writing (when IWC is to collocate with a conference) and coordination with the confluence competition (CoCo). The SC is responsible for archiving the (informal) workshop proceedings.
- 9 Any modification of the bylaws should be proposed to the SC four weeks before the workshop and ratified at the meeting.

## Competitions (Selection)

► CASC

► CoCo

► termCOMP



## Competitions (Selection)

► CASC

► CoCo

► termCOMP



## Competitions (Selection)

- ▶ CASC
- ▶ CoCo
- ▶ termCOMP

## Confluence Competition

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <https://project-coco.uibk.ac.at/>
- ▶ friendly competition with clear **bylaws**

## Competitions (Selection)

- ▶ CASC
- ▶ CoCo
- ▶ termCOMP

## Confluence Competition

- ▶ established in 2012 (29 May, Nagoya)
- ▶ permanent website: <https://project-coco.uibk.ac.at/>
- ▶ friendly competition with clear **bylaws**

## Bylaws

- 1 The steering committee (SC) is responsible for all affairs concerning the confluence competition:
  - ▶ deciding the date and location of the next competition,
  - ▶ deciding the categories in the next competition,
  - ▶ selecting panel members (who provide secret digits that determine the problems selected for the next competition),

## Bylaws (cont'd)

- ①
  - ▶ running the live competition and exporting the results,
  - ▶ collecting one-page tool descriptions for inclusion in the proceedings of IWC,
  - ▶ performing a full run with corrected tools within one month of the live competition and exporting the tools to CoCoWeb,
  - ▶ maintaining the infrastructure of the confluence competition, including COPS and CoCoWeb,
  - ▶ publicity.
- ② SC members must have been involved in a tool that participated in a past competition.
- ③ SC members can serve at most two consecutive terms. A term consists of four consecutive competitions.
- ④ The SC selects its own chair.
- ⑤ Decisions in the SC (if not by global agreement) are by simple majority vote. In case of a draw the chair decides.

## SOME (RANDOM) REMARKS

- rules of the competition ?
- what is the aim of the termination competition ?
- who will win next year's competition ?
- why should  
Cime, Matchbox, MultumNonMultu, TEPARLA, TPA, TTT, TTTbox, ...  
participate ?
- new TPDB problems ?
- **termtools** mailing list not working:  
no discussion, no agreement, no archive  $\Rightarrow$  no change

# Outline

1. Hans Zantema
2. WST & IWC
3. Hans Zantema
4. Workshops and Competitions
- 5. Hans Zantema**



*“Het licht schijnt in de duisternis  
en de duisternis heeft het niet kunnen doven”  
Joh. 1:5*

Diepbedroefd zijn wij omdat van ons is heengegaan onze geliefde,  
eigenwijze echtgenoot, vader en schoonvader.  
Net als hijzelf, kijken wij met dankbaarheid terug op zijn leven.

Hij was wetenschapper in hart en nieren:

## Prof. Dr. Hans Zantema Hantsje

Goingarijp  
11 juni 1956

Eindhoven  
28 januari 2025