

# Confluence of Right Ground Term Rewriting Systems is Decidable

*FOSSACS 2005*

Łukasz Kaiser

Mathematische Grundlagen der Informatik  
RWTH Aachen

# Outline

---

- Introduction
- Basic reductions of rewriting systems
- Coloured rewriting
- Automata with constraints
- Conclusions

# Terms

---

- Constant signature with arity
- Positions in terms, variables
- Substitutions

$$f(x, g(c, y))$$

- ground terms
- linear terms

# Rewriting Systems

---

- Rewrite rules

$$l \rightarrow r$$

$$f(x, g(c, y)) \rightarrow g(x, y)$$

$$f(x, x) \rightarrow c$$

- TRS - set of rewrite rules

- $\xrightarrow{*}$  - transitive reflexive closure of  $\rightarrow$

- syntactic classes of TRS

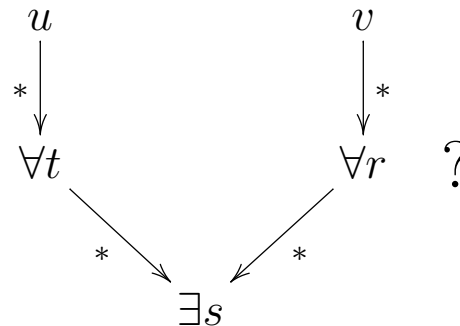
- ground

- right ground

- left linear right ground

# Problems in TRS

- reachability:  $u \xrightarrow{*} v$  ?
- joinability:  $u \xrightarrow{*} \exists t \xleftarrow{*} v$  ?
- deep joinability:



- confluence:  $u$  deep joinable with  $u$  ?
- TRS confluence: all  $u$  confluent
- Normal form problems

# Methods and Results

---

- Ground TRS
  - polynomial time
  - transitive closure, automata
- Left linear right ground TRS
  - up to exponential
  - tree transducers
  - confluence in coNP - open
- Right ground TRS
  - decidable [Tiwari, Godoy, Verma], elementary
  - rewrite closure, automata with constraints
- Further syntactic classes
  - shallow rules, else gets undecidable very quickly

# TRS Reductions

# Naming with Constants

---

Change

$$l \rightarrow f(c_1, c_2)$$

to

$$l \rightarrow c_{new}$$

$$c_{new} \rightarrow f(c_1, c_2)$$

Rules left:

$>$ : rules in the form  $c \rightarrow f(c_1, \dots, c_n)$ ,

$\leq$ : rules  $t \rightarrow c$ , where  $t$  is any term.



# Normalized RGTRS

---

*Lemma:* reachability for RGTRS is decidable.

Use this lemma to compute closure of  $\rightarrow$  up to terms of height 1 in an RGTRS after naming constants.

*Lemma:*

$$t := f(t_1, \dots, t_n) \xrightarrow{*} s$$

for ground terms iff

- (1)  $s = f(s_1, \dots, s_n)$  and for each  $i$  we have  $t_i \xrightarrow{*} s_i$ ,
- (2) there is a constant  $c$  such that  $t \xrightarrow{*} c$  and  $c \xrightarrow{*}_{>} s$ .

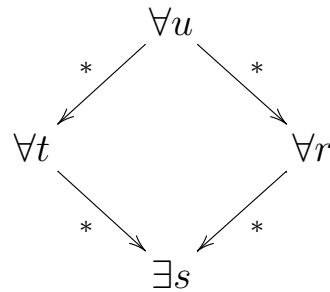
# Stability

---

- *Root - Stable*: does not rewrite to any constant
- *Stable*:
  - all subterms root - stable,
  - no subterm rewrites to a constant
  - all successors in  $\xrightarrow{*}$  can be reached by  $\xrightarrow{*}>$

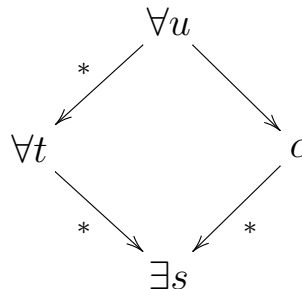
# Reducing Confluence

Confluence in a normalized RGTRS:



reduces to:

- Deep joinability of constants
- Semi confluence



# Coloured Rewriting

# Rewrite Closure

---

Idea:

$$\xrightarrow{*} = \rightarrow_1 \circ \rightarrow_2$$

where  $\rightarrow_1$  and  $\rightarrow_2$  might not be defined as rewriting relations but are in some way easier to analyze

More specifically it is used with

- $\rightarrow_1$  - constrained decreasing rewriting ( $\leq$ )
- $\rightarrow_2$  - increasing ground rewriting ( $>$ )

# Constrained Rewriting

---

$$l \rightarrow r \text{ if } [condition]$$

Conditions generally on reachability and joinability on variables from  $l$ :

$$x|y$$

$$c \xrightarrow{*} z$$

$$f(x, g(y, z)) \rightarrow c \text{ if } [x|y, d \xrightarrow{*} z]$$

# Colour Constraints

---

- Ad-hoc constrained rewriting where constraints just specify reachability from constants
- *Colour*  $K$  is a set of constants  $K = \{c_1, \dots, c_m\}$
- Ground term  $t$  has colour  $K$  if each  $c_i \xrightarrow{*} t$ .
- Each ground term  $t$  has one biggest colour

$$K(t) := \{c : c \xrightarrow{*} t\}$$

# Coloured Terms and Rewrite Rules

---

- Term  $t$  with variables with assigned colours,  
 $K(x)$  required for  $x$
- Correct ground substitutions  $\sigma$  substitutes  $s$  for  $x$  only  
if  $K(x)$  is a colour of  $s$
- Coloured rewrite rules
- Coloured (constrained) rewriting



# Propagating Colours

---

Take term  $t$  with a colour constraint:

$$t = f(x, y), \quad c \in K(t)$$

What are the colour constraints for  $x, y$  that ensure this?

As the rewriting system is normalized take all

$$c \rightarrow f(c_1, c_2)$$

and pairs of constraints

$$K(x) = c_1, \quad K(y) = c_2$$

Note: more than one resulting colour constraint

# Reducing $\succ$ Rewrites (1)

---

- Think about  $t \xrightarrow{*}_{>} s \rightarrow c$

- Take the rule

$$l \rightarrow c$$

- Cut  $l$  at some positions and put there constants

- Grow these constants with  $\xrightarrow{*}_{>}$  back to  $l$  size

- Check what colours must be put on variables

# Reducing > Rewrites (2)

---

Example: take TRS

$$R = \{c \rightarrow f(c, c), f(x, f(x, x)) \rightarrow c\}$$

and look at rewriting:

$$f(c, c) \rightarrow f(c, f(c, c)) \rightarrow c$$

This suggests to cut and grow:

$$l = f(x, f(x, x)) \text{ cut } f(x, c) \rightarrow_{>} f(x, f(c, c))$$

New coloured rule:  $f(x : \{c\}, c) \rightarrow c$

Note: colour constraints necessary since terms non-linear

# Coloured Unification

---

Problem:

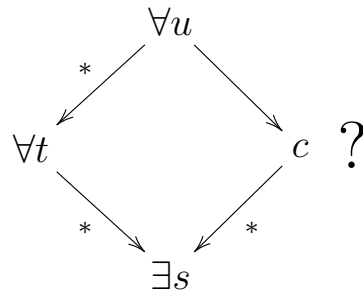
rewrite  $t$  with  $l \rightarrow r$

when both  $t$  and  $l$  coloured and with variables

- unify  $t$  and  $l$  in the standard way
- propagate colours to satisfy constraints on substituted variables
- extend constraints on the variables that are left
- return a finite set of coloured "most general" unifiers

# Reducing Semi Confluence

When not



- Track coloured rewrites  $u \xrightarrow{*} t$
- Find all possible coloured mgus for  $t$
- Check stability of such  $t$  and non-joinability with  $c$

# Tree Automata with Constraints

# Definition

---

- Automata with Equality and Disequality Constraints (AWEDC)
- *Equality (disequality) constraint* is  $p_1 = p_2$  ( $p_1 \neq p_2$ ), where  $p_1$  and  $p_2$  are positions in terms
- Transition rules  $f(q_1, \dots, q_n) \rightarrow^\alpha q$ ,  $\alpha$  is a boolean combination of equality and disequality constraints
- Reduction automata: there is an ordering on states so if

$$f(q_1, \dots, q_n) \rightarrow^\alpha q$$

and  $\alpha \neq \emptyset$  then  $q$  is strictly smaller than each  $q_i$

# Properties

---

- The emptiness of a language accepted by a reduction automaton is decidable.
- The class of reduction automata is closed under union and intersection. There is a construction for the union that preserves determinism.
- With each reduction automaton we can associate a complete reduction automaton that accepts the same language. This construction preserves determinism.
- The class of complete deterministic reduction automata is closed under complement.



# Standard Use - Normal Forms

---

- Problem: automata for normal forms
- Standard tree automata for linear rules, constraints needed for  $f(x, x) \rightarrow c$
- There is a (deterministic) reduction automata that accepts substitutions for a term  $t$
- Possible to extend this to accept all terms encompassing such substitutions

# Reductions (1)

---

- Extend the construction for normal forms to take colours into account
  - colours grow from constants so are checkable by standard tree automaton
- Also need to guarantee that the result is not joinable with  $c$
- Again use normalization of the TRS and standard automata to check it

# Reductions (2)

---

Deep joinability of constants remains to be checked

- Reduce it to emptiness of reduction automata
- Need to extend signature to operate on pairs of terms
  - similar to transducers with additional marking
- Normalization of TRS also necessary

# Conclusions

---

- Rewrite closure can help a lot
- Constrained rewriting in different flavours is useful
- Don't forget about automata with constraints
  - when working on transducers, regular structures
  - make things more expressive
  - need more care by intersection + complementation

# Thank you!