Confluence of Right Ground Term Rewriting Systems is Decidable

FOSSACS 2005

Łukasz Kaiser

Mathematische Grundlagen der Informatik

RWTH Aachen

GAMES

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

$$l \to r$$

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

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

- TRS - set of rewrite rules
- $\xrightarrow{*}$ - transitive reflexive closure of $\to$
- 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:

$$\begin{tikzcd}
u \arrow[d, "*"'] & & v \arrow[d, "*"] & \\
\forall t \arrow[dr, "*"'] & & \forall r \arrow[dl, "*"] & ? \\
& \exists s & &
\end{tikzcd}$$

- 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

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

to

$$l \to c_{new}$$

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

Rules left:

$>$: rules in the form $c \to f(c_1, \ldots, c_n)$,

$\leq$: rules $t \to c$, 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:

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

for ground terms iff

(1) $s = f(s_1, \ldots, s_n)$ 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:

$$\begin{tikzcd} & \forall u \arrow[dl, "*"'] \arrow[dr, "*"] & \\ \forall t \arrow[dr, "*"'] & & \forall r \arrow[dl, "*"] \\ & \exists s & \end{tikzcd}$$

reduces to:

- Deep joinability of constants
- Semi confluence

$$\begin{tikzcd} & \forall u \arrow[dl, "*"'] \arrow[dr] & \\ \forall t \arrow[dr, "*"'] & & c \arrow[dl, "*"] \\ & \exists s & \end{tikzcd}$$

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

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

Conditions generally on reachability and joinability on variables from l:

$$x|y$$

$$c \xrightarrow{*} z$$

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

Colour Constraints

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

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

GAMES

Coloured Terms and Rewrite Rules

- Term t with variables with assigned colours, $K(x)$ required for x
- Correct ground substitutions σ 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:

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

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

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

Note: more than one resulting colour constraint

Reducing $>$ Rewrites (1)

- Think about $t \xrightarrow{*}_{>} s \to c$
- Take the rule

$$l \to c$$

- 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

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

and look at rewriting:

$$f(c,c) \to f(c, f(c,c)) \to c$$

This suggests to cut and grow:

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

New coloured rule: $f(x : \{c\}, c) \to c$
Note: colour constraints necessary since terms non-linear

GAMES

Coloured Unification

Problem:

$$\text{rewrite } t \text{ with } l \to r$$

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

$$\begin{tikzcd} & \forall u \arrow[dl, "*"'] \arrow[dr] & \\ \forall t \arrow[dr, "*"'] & & c \quad ? \arrow[dl, "*"] \\ & \exists s & \end{tikzcd}$$

- 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 $f(q_1, \ldots, q_n) \rightarrow^{\alpha} q$, α is a boolean combination of equality and disequality constraints
- Reduction automata: there is an ordering on states so if

$$f(q_1, \ldots, q_n) \rightarrow^{\alpha} q$$

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 $f(x, x) \to c$
- 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 accout
  - 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!