Term Rewriting Systems

Franz Baader
Theoretical Computer Science
TU Dresden
Germany

1. Motivation and basic definitions and results.
2. Equational Problems: the word problem and term rewriting
3. Termination of term rewriting systems
4. Confluence of term rewriting systems
5. Completion of term rewriting systems

Confluence

The TRS R is confluent iff $\rightarrow_R$ is confluent.

Theorem (undecidability of confluence)

The following problem is in general undecidable:

Given: A finite TRS R.

Question: Is R confluent or not?

Proof by reduction of the word problem:

Let E be a finite set of identities $l \approx r$ such that $\mathcal{V}ar(l) = \mathcal{V}ar(r)$ and neither l nor r are variables (without loss of generality!).

Given ground terms s, t we define

$$R_{s,t} := E \cup E^{-1} \cup \{a \rightarrow s, a \rightarrow t\}$$

where a is a new constant.

Then $R_{s,t}$ is confluent iff $s \approx_E t$.

An important decidable case

terminating TRS

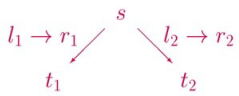
Theorem

If $\rightarrow_R$ is locally confluent and terminating, then it is also confluent.

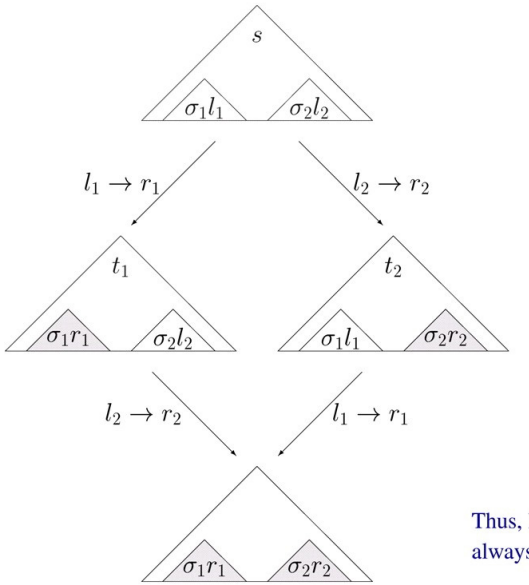
local confluence

In the following, we assume that R is terminating.

To test for local confluence of $\rightarrow_R$ we consider the reductions

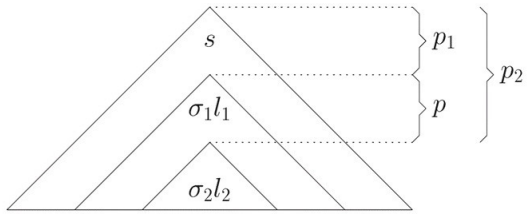


Case 1: Reductions in separate subtrees:

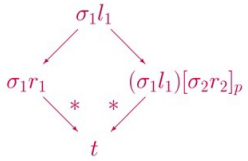


Thus, local confluence always holds in this case.

Case 2: Reductions in overlapping subtrees:



We can now restrict our attention to the subterm $s|_{p_1} = \sigma_1 l_1$ because

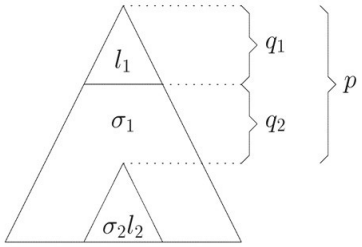


implies

$$t_1 = s[\sigma_1 r_1]_{p_1} \overset{*}{\to} s[t]_{p_1} \overset{*}{\leftarrow} s[(\sigma_1 l_1)[\sigma_2 r_2]_p]_{p_1} = s[\sigma_2 r_2]_{p_2} = t_2$$

Dresden

Case 2.1: Overlap in substitution part (non-critical overlap):

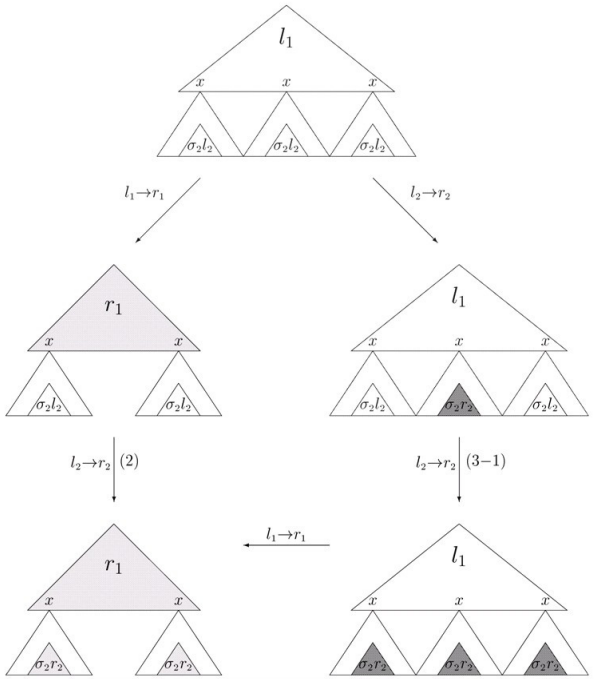


Local confluence holds in this situation as well.

The analysis is complicated by the fact that $x := l_1|_{q_1}$ can occur repeatedly in both l_1 and r_1.

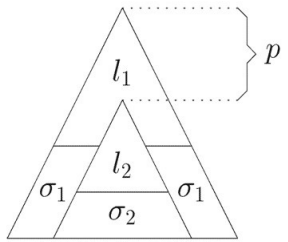
We consider a prototypical situation where x occurs three times in l_1 and twice in r_1.

Dresden



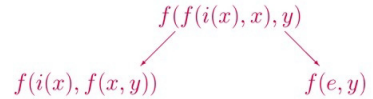
Dresden

Case 2.2: Overlap of the left-hand sides (critical overlap):



In this case, local confluence need not hold.

$$\begin{aligned} f(f(x,y),z) &\to f(x,f(y,z)), \\ f(i(x),x) &\to e, \end{aligned}$$



Thus, such critical overlaps must explicitly be checked for local confluence.

Fortunately, it is enough to check finitely many critical pairs.

Dresden

Unification

basic definitions and results

Definition

A unifier of the terms s, t is a substitution σ such that $\sigma(s) = \sigma(t)$.

A substitution σ is more general than a substitution σ' if there is a substitution δ such that

$$\sigma' = \delta\sigma.$$

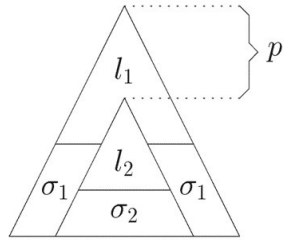
In this case we write $\sigma \lesssim \sigma'$, and say that σ' is an instance of σ.

Definition

The unifier σ of the terms s, t is a most general unifier (mgu) iff every unifier of s, t is an instance of σ.

We will show: if s, t have a unifier, then they have an mgu, and this mgu can effectively be computed.

Case 2.2: Overlap of the left-hand sides (critical overlap):



We have

$$\sigma_1(l_1|_p) = \sigma_1(l_1)|_p = \sigma_2(l_2).$$

If we assume (without loss of generality) that the rules

$$l_1 \to r_1 \text{ and } l_2 \to r_2$$

have no variables in common, then this implies that there is a single substitution σ such that

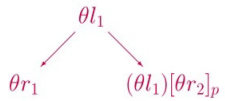
$$\sigma(l_1|_p) = \sigma(l_2).$$

Definition (critical pair)

Let $l_1 \to r_1, l_2 \to r_2$ be two rules whose variables have been renamed such that $\mathcal{V}ar(l_1, r_1) \cap \mathcal{V}ar(l_2, r_2) = \emptyset$.

Let $p \in \mathcal{P}os(l_1)$ be such that $l_1|_p$ is not a variable, and let θ be an mgu of $l_1|_p$ and l_2.

This determines a critical pair $\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$:



The critical pairs of R are the critical pairs between any two of its (renamed) rules and are denoted by $CP(R)$.

This includes overlaps of a rule with a renamed variant of itself!

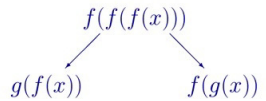
Example

$$f(f(x)) \to g(x)$$

$$f(f(y)) \to g(y)$$

The lhs $f(f(x))$ unifies with the subterm $f(y)$ of the renamed lhs producing the mgu $\{y \mapsto f(x)\}$.

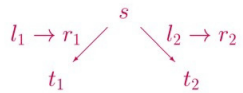
Thus we obtain the critical pair



This shows that $R := \{f(f(x)) \to g(x)\}$ is not confluent, because the terms in the critical pair are not joinable.

Lemma (critical pair lemma)

If



then $t_1 \downarrow_R t_2$ or

$$t_1 = s[u_1]_{p_1} \quad \text{and} \quad t_2 = s[u_2]_{p_1},$$

where $\langle u_1, u_2 \rangle$ or $\langle u_2, u_1 \rangle$ is an instance of a critical pair of R.

If there is no overlap or a non-critical overlap, then $t_1 \downarrow_R t_2$.

Otherwise, $s|_{p_1} = \sigma(l_1)$ and $\sigma(l_1|_p) = \sigma(l_2)$.

Thus, σ is a unifier of $l_1|_p$ and l_2, which shows that it is an instance of the mgu used when computing the critical pair.

Theorem (critical pair theorem)

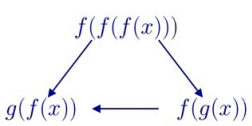
A TRS is locally confluent iff all its critical pairs are joinable.

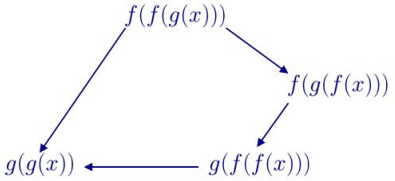
Example

The TRS

$$\{f(f(x)) \to g(x), \quad f(g(x)) \to g(f(x))\}$$

is locally confluent.





Corollary

Confluence of a finite and terminating TRS R is decidable.

Proof:

Since R is terminating, $\to_R$ is confluent iff it is locally confluent.

There are only finitely many critical pairs.

Joinability of each critical pair can effectively be tested since R is terminating.

Example

The TRS $\{f(f(x)) \to g(x), \quad f(g(x)) \to g(f(x))\}$ is locally confluent.

Since it is also terminating, it is thus confluent.

use LPO with $f > g$

Unification

in more detail

Definition

A unification problem is a finite set of equations

$$S = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}.$$

A unifier or solution of S is a substitution σ such that $\sigma s_i = \sigma t_i$ for $i = 1, \ldots, n$.

$\mathcal{U}(S)$ denotes the set of all unifiers of S.

A substitution σ is a most general unifier (mgu) of S if

- $\sigma \in \mathcal{U}(S)$ and
- $\forall \sigma' \in \mathcal{U}(S).\ \sigma \lesssim \sigma'$.

Examples

some typical situations

$\{f(x) =^? f(a)\}$	has $\{x \mapsto a\}$ as a unifier
$\{x =^? f(y)\}$	has $\{x \mapsto f(y)\}$, $\{x \mapsto f(a), y \mapsto a\}, \ldots$ as unifiers
$\{x =^? f(x)\}$	does not have a unifier
$\{f(x) =^? g(y)\}$	does not have a unifier

Dresden

Unification

by transformation

Idea:

Transform set of equations into solved form, from which the mgu can be obtained immediately.

Similar to Gaussian elimination in linear algebra:

$$\begin{array}{|rcl|} \hline x + 3y &=& 0 \\ 2x + 8y &=& 2z \\ \hline \end{array} \quad \rightsquigarrow \quad \begin{array}{|rcl|} \hline x + 3y &=& 0 \\ 2y &=& 2z \\ \hline \end{array} \quad \rightsquigarrow \quad \begin{array}{|rcl|} \hline x + 3y &=& 0 \\ y &=& z \\ \hline \end{array}$$

$$\rightsquigarrow \quad \begin{array}{|rcl|} \hline x + 3z &=& 0 \\ y &=& z \\ \hline \end{array} \quad \rightsquigarrow \quad \begin{array}{|rcl|} \hline x &=& -3z \\ y &=& z \\ \hline \end{array}$$

Dresden

Definition

A unification problem

$$S = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$$

is in solved form if the x_i are pairwise distinct variables, none of which occurs in any of the t_i.

In this case we define

$$\vec{S} := \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}.$$

Lemma

If S is in solved form then $\vec{S}$ is an mgu of S.

Dresden

The transformation rules

Delete	$\{t =^? t\} \uplus S$	$\Longrightarrow$	S
Decompose	$\{f(t_1, \ldots, t_n) =^? f(u_1, \ldots, u_n)\} \uplus S$	$\Longrightarrow$	$\{t_1 =^? u_1, \ldots, t_n =^? u_n\} \cup S$
Orient	$\{t =^? x\} \uplus S$	$\Longrightarrow$	$\{x =^? t\} \cup S$ if $t \notin V$
Eliminate	$\{x =^? t\} \uplus S$	$\Longrightarrow$	$\{x =^? t\} \cup \{x \mapsto t\}(S)$ if $x \in \mathcal{V}ar(S) - \mathcal{V}ar(t)$

Dresden

Example

$\{x \stackrel{?}{=} f(a),\ g(x,x) \stackrel{?}{=} g(x,y)\}$ $\Longrightarrow_{\text{Eliminate}}$

$\{x \stackrel{?}{=} f(a),\ g(f(a),f(a)) \stackrel{?}{=} g(f(a),y)\}$ $\Longrightarrow_{\text{Decompose}}$

$\{x \stackrel{?}{=} f(a),\ f(a) \stackrel{?}{=} f(a),\ f(a) \stackrel{?}{=} y\}$ $\Longrightarrow_{\text{Delete}}$

$\{x \stackrel{?}{=} f(a),\ f(a) \stackrel{?}{=} y\}$ $\Longrightarrow_{\text{Orient}}$

$\{x \stackrel{?}{=} f(a),\ y \stackrel{?}{=} f(a)\}$

$Unify(S)$ = `while` there is some T such that $S \Longrightarrow T$ `do` S `:=` T`;`
`if` S is in solved form `then return` $\vec{S}$ `else fail.`

Lemma (termination, soundness, completeness)

1. *Unify* terminates for all inputs.
2. If $S \Longrightarrow T$ then $\mathcal{U}(S) = \mathcal{U}(T)$.
3. If $Unify(S)$ returns σ, then σ is an mgu of S.
4. If $Unify(S)$ fails, then the final set of equations contains an equation of the form
   (a) $x \stackrel{?}{=} t$ with $x \in \mathcal{V}ar(t), x \neq t$,
   (b) $f(\cdots) \stackrel{?}{=} g(\cdots)$ with $f \neq g$.
5. If $Unify(S)$ fails, then S has no solution.

Theorem

The function *Unif* decides, for any input unification problem S, whether it has a solution or not.

If S has a solution, then *Unif* computes an mgu of S.

Complexity:

- The worst-case complexity of this unification algorithm is exponential (both time and space).
- There exists a linear time unification algorithm.

Matching

a special case of unification

Given terms l, s, find a substitution σ such that $\sigma(l) = s$.
The substitution σ is called a matcher of the matching problem $l \lesssim^{?} s$.

Reduce matching to unification: regard all variables in s as constants, by introducing a new constant c_x for each variable x.

Example

The matching problem $f(x,y) \lesssim^{?} f(g(z),x)$

becomes the unification problem $\{f(x,y) \stackrel{?}{=} f(g(c_z),c_x)\}$.

The mgu $\{x \mapsto g(c_z), y \mapsto c_x\}$

becomes the matcher $\{x \mapsto g(z), y \mapsto x\}$.