Term Rewriting Systems

Ferucio Laurențiu Țiplea
Visiting Professor
School of Computer Science
University of Central Florida
E-mail: `tiplea@cs.ucf.edu`

November 18, 2004

Contents:

Bibliography:

1. Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science 55, Cambridge University Press, 2003 (884 pages)

2. F. Baader, T. Nipkow. Term Rewriting and All That, Cambridge University Press, 1998 (301 pages)

Motivations

Syntactically, rewrite rules are a special kind of equations that can be applied in one direction only.

A term rewriting system (trs, for short) is a set of rewrite rules. They have many applications to:

- theorem proving
- algebraic specification (of data types, programs etc.)
- computer algebra
- λ-calculus
- implementation of declarative languages
- operational semantics of programming languages

Motivations

Example: Ackerman-Peter function $f : \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$

(R1) $f(0, y) = y + 1$

(R2) $f(x + 1, 0) = f(x, 1)$

(R3) $f(x + 1, y + 1) = f(x, f(x + 1, y))$

for all $x, y \in \mathbf{N}$.

A few values of this function:

- $f(0, y) = y + 1$
- $f(1, y) = y + 2$
- $f(2, y) = 2y + 3$

for all $y \in \mathbf{N}$.

Example of computation:

$$f(2,1) \overset{(R3),\ x\to 1,\ y\to 0}{=} f(1,f(2,0))$$

$$f(1,f(2,0)) \overset{(R2),\ x\to 1}{=} f(1,f(1,1))$$

$$f(1,f(1,1)) \overset{(R3),\ x\to 0,\ y\to 0}{=} f(1,f(0,f(1,0)))$$

$$f(1,f(0,f(1,0))) \overset{(R2),\ x\to 0}{=} f(1,f(0,f(0,1)))$$

$$f(1,f(0,f(0,1))) \overset{(R1),\ y\to 1}{=} f(1,f(0,2))$$

$$f(1,f(0,2)) \overset{(R1),\ y\to 2}{=} f(1,3)$$

$$\ldots$$

Motivations

Conclusions:

- each element of this computation is a term
- each computation step is based on applying one of the equations (R1), (R2) or (R3)
- each equation is used in one direction only ("from left to right")
- each equation is based on a substitution ("$x \to 1$, $y \to 0$") which matches the left hand side of the equation to some subterm of the current term
- the immediate successor of a term t is obtained by replacing a subterm of t by an instance of the right hand side of some equation

Terms and Term Rewriting Systems

Let $\mathcal{F}$ be a set of function symbols each of which having associated an arity, and let X be a set of variables. Assume that $\mathcal{F}$ and X are disjoint sets. The set of terms over $\mathcal{F}$ and X is defined inductively as follows:

- each function symbol of arity 0 is a term;
- each variable is a term;
- if $t_1, \ldots, t_n$ are terms and f is a function symbol of arity $n \geq 1$, then $f(t_1, \ldots, t_n)$ is a term.

Function symbols of arity 0 are usually called constant symbols.

Denote by $T(\mathcal{F}, X)$ the set of all terms over $\mathcal{F}$ and X.

Examples of terms:

- x is a term, for any variable x
- a is a term, for any constant symbol a
- $f(x, x)$ is a term, where f is a function symbol of arity 2
- $f(f(x, x), a)$ is a term
- $f(g(a), f(f(x, x), a))$ is a term, where g is a function symbol of arity 1
- all expressions in the computation

  $$f(2, 1) = \cdots = f(1, 3)$$

  of the Ackerman-Peter function are terms

Variables in a term t

Given a term t, we denote by $Var(t)$ the set of all variables occurring in t. If $Var(t) = \emptyset$ then t is called a ground term.

Example: if $t = f(x, g(y, x), z)$, then $Var(t) = \{x, y, z\}$

Subterms

Given a term t, we denote by $Sub(t)$ the set of all subterms of t.

Example: if $t = f(x, g(y, x), z)$, then $Sub(t) = \{t, x, y, z, g(y, z)\}$

Terms and Term Rewriting Systems

Rewrite rule: a pair of terms $r = (t_1, t_2)$, also written as $r: t_1 \to t_2$, such that

- t_1 is not a variable
- $Var(t_2) \subseteq Var(t_1)$

t_1 (t_2, resp.) is usually called the left hand side (right hand side, resp.) of r and it is denoted by $lhs(r)$ ($rhs(r)$, resp.).

Example:

- $f(x+1, 0) \to f(x, 1)$ is a rewrite rule
- neither $x \to f(a, a)$ nor $f(x, y) \to f(0, z)$ is a rewrite rule

A non-empty set of rewrite rules is called a term rewriting system.

Substitution: function from X into $T(\mathcal{F}, X)$

Example: $\sigma : X \rightarrow T(\mathcal{F}, X)$ given by $\sigma(x) = f(x,x)$, $\sigma(y) = a$ and $\sigma(z) = z$, for all $z \neq x$ and $z \neq y$.

Substitutions can be applied to terms. They substitute all variables but leave unchanged all function symbols.

Formally, each substitution $\sigma : X \rightarrow T(\mathcal{F}, X)$ is extended to a homomorphism from $T(\mathcal{F}, X)$ to $T(\mathcal{F}, X)$, which is also denoted by σ.

Example:

- $\sigma(f(x,x)) = f(\sigma(x),\sigma(x)) = f(f(x,x),f(x,x))$
- $\sigma(f(x,g(y,x),z)) = f(\sigma(x),g(\sigma(y),\sigma(x)),\sigma(z)) = f(f(x,x),g(a,f(x,x)),z)$

The domain of a substitution σ is

$$Dom(\sigma) = \{x \in X | \sigma(x) \neq x\}$$

If $Dom(\sigma)$ is finite, $Dom(\sigma) = \{x_1,\ldots,x_n\}$, we may write σ as a set

$$\sigma = \{x_1 \to \sigma(x_1),\ldots,x_n \to \sigma(x_n)\}$$

In such a case, $\sigma(t)$ is usually writen as

$$t[x_1/\sigma(x_1),\ldots,x_n/\sigma(x_n)]$$

Unification

A substitution σ is called a unifier of two terms t_1 and t_2 if $\sigma(t_1) = \sigma(t_2)$. Moreover, t_1 and t_2 are called unifiable.

Example:

- let $\sigma : X \rightarrow T(\mathcal{F}, X)$ given by $\sigma(x) = a$, $\sigma(y) = a$ and $\sigma(z) = z$, for all $z \neq x$ and $z \neq y$
- let $t_1 = f(x, x)$ and $t_2 = f(a, a)$
- σ is a unifier of t_1 and t_2
- let $t_3 = f(a, b)$, where $b \neq a$
- σ is not a unifier of t_1 and t_3

Rewriting

Let R be a trs. Define a binary relation on terms, $\Rightarrow_R$, as follows:

$$t_1 \Rightarrow_R t_2$$

iff

- $t_1 = u\, t_0\, v$, where the decomposition ut_0v means that t_0 is a subterm of t_1
- there exist a rule $r : t \rightarrow t' \in R$ and a unifier σ of t_0 and t
- $t_2 = u\, \sigma(t')\, v$

$\stackrel{+}{\Rightarrow}_R$ is the transitive closure, and $\stackrel{*}{\Rightarrow}_R$ is the reflexive and transitive closure, of $\Rightarrow_R$

Example: Let $R = \{r_1 : f(0,y) \rightarrow y+1,\ r_2 : f(x+1,0) \rightarrow f(x,1),\ r_3 : f(x+1,y+1) \rightarrow f(x, f(x+1,y))\}$. Then,

- $f(2,1) \Rightarrow_R f(1, f(2,0))$
- $f(1, f(2,0)) \Rightarrow_R f(1, f(1,1))$
- $f(1, f(1,1)) \Rightarrow_R f(1, f(0, f(1,0)))$
- $f(1, f(0, f(1,0))) \Rightarrow_R f(1, f(0,2))$

Therefore,

$$f(2,1) \stackrel{*}{\Rightarrow}_R f(1, f(0,2))$$

Canonical Form

Let R be a trs.

- R is called terminating or noetherian if there is no infinite sequence of terms

  $$t_1, t_2, \ldots$$

  such that $t_i \Rightarrow_R t_{i+1}$, for all $i \geq 1$;

- R is called confluent or Church-Rosser if

  $$(\forall t, t_1, t_2)(t \overset{*}{\Rightarrow}_R t_1 \wedge t \overset{*}{\Rightarrow}_R t_2 \Rightarrow (\exists t')(t_1 \overset{*}{\Rightarrow}_R t' \wedge t_2 \overset{*}{\Rightarrow}_R t'))$$

- R is called canonical or complete if there it is terminating and confluent

Canonical Form

Exercise: Let $R = \{r_1 : \ f(0, y) \to y+1, \ r_2 : \ f(x+1, 0) \to f(x, 1), \ r_3 : \ f(x+1, y+1) \to f(x, f(x+1, y))\}$.

Prove that R is a canonical trs

Hint: By mathematical induction on x and y

Canonical Form

Let R be a trs and t a term. Then,

- t is called irreducible or a normal or reduced form under R if there is not t' such that $t \Rightarrow_R t'$
- t' is called a normal or reduced form of t under R if $t \overset{*}{\Rightarrow}_R t'$ and t' is a normal form

Example: Let $R = \{r_1 : f(0, y) \to y+1,\ r_2 : f(x+1, 0) \to f(x, 1),$ $r_3 : f(x+1, y+1) \to f(x, f(x+1, y))\}$. Then,

- 2 is a normal form
- $f(1, 0) \Rightarrow_R f(0, 1) \Rightarrow_R 2$ and, therefore, 2 is a normal form of $f(1, 0)$

Canonical Form

Theorem 1 If R is a canonical term rewriting system, then any term t has a unique normal form.

Proof (Sketch) Each term has at least a normal form by the termination property.

If t is a term and t_1 and t_2 are normal forms of t, then $t_1 = t_2$ by confluence. □

The unique normal form of a term t under a canonical trs R is called the canonical form of t under R, and it is denoted by $\|t\|_R$.

Canonical Form

Why canonical forms are important?

Theorem 2 If R is a canonical term rewriting system, then

$$R \models t_1 = t_2 \quad \Leftrightarrow \quad \|t_1\|_R = \|t_2\|_R,$$

for any terms t_1 and t_2 ($R \models t_1 = t_2$ means that the equation $t_1 = t_2$ can be deduced from the equations in R).

The theorem above provides us with a very natural procedure for deciding the equality of two terms: we can decide whether or not t_1 and t_2 can be proved equal using the equations in R by checking whether their canonical forms are identical.

Proving Termination

Theorem 3 The following problem is undecidable:

Instance: finite term rewriting system R and a term t
Question: are all computations starting with t terminating?

Proof (Sketch)

Reduce the halting problem for Turing machines to this problem:

Instance: Turing machine M and input w
Question: does M halt on w?

(associate to M a trs R_M and to each configuration C a term t_C such that $C \vdash_M C' \quad \Leftrightarrow \quad t_C \Rightarrow_{R_M} t_{C'}$) □

Theorem 4 The following problem is undecidable:

Instance: finite term rewriting system R
Question: is R terminating?

Proof (Sketch)

Reduce the uniform halting problem for Turing machines to this problem:

Instance: Turing machine M
Question: does M halt on all inputs?

□

A trs R is called right-ground if each rewrite rule $t_1 \rightarrow t_2 \in R$ satisfies $Var(t_2) = \emptyset$.

Theorem 5 Let R be a right-ground trs. Then, R does not terminate if and only if there exists a rule $t_1 \rightarrow t_2 \in R$ such that $t_2 \stackrel{+}{\Rightarrow}_R ut_2v$ (i.e., t_2 is a subterm of ut_2v).

Corollary 1 Termination for finite right-ground trs is decidable.

Decision procedure for the termination of right-ground trs

1. consider all right hand sides of the rewrite rules in R, and simultaneously generate all reduction sequences starting with these terms;

2. if R does not terminate then there exists a right hand side t_2 which generate ut_2v for some u and v, where t_2 is a subterm in ut_2v. Moreover, ut_2v is obtained after finitely many steps;

3. if R terminates then all computation trees are finite and they can be obtained after finitely many steps.

Proving Termination

Therefore, after finitely many steps

- either we get a term ut_2v for some u and v, where t_2 is a subterm in ut_2v (and in this case R does not terminate),
- or all computation trees associated to the right hand sides of the rewrite rules in R are finite (and in this case R is terminating).

Example: Let $R = \{f(x,x) \rightarrow g(a),\ g(x) \rightarrow f(g(a),b)\}$. Then,

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

which shows that R is not terminating.

Proving Termination

Techniques for proving termination (see [1] for details):

1. semantic methods - based on suitable interpretations
   (a) well-founded monotone algebras
   (b) polynomial interpretations
2. syntactic methods - based upon orders on terms
   (a) recursive path order
   (b) Knuth-Bendix order
3. transformational methods - based on applying transformations to term rewriting systems
   (a) dummy elimination
   (b) semantic labeling
   (c) abstract commutations

Proving Confluence

Theorem 6 The following problem is undecidable:

Instance: finite term rewriting system R
Question: is R confluent?

Proof (Sketch)

Reduce the word problem to this problem:

Instance: set E of equations
Question: does $t_1 = t_2$ can be deduced from E, $\forall t_1, t_2$?

(associate to E a trs R_E such that the word problem for E is decidable iff R is confluent) □

Proving Confluence

A trs R is locally confluent if

$$(\forall t, t_1, t_2)(t \Rightarrow_R t_1 \ \wedge \ t \Rightarrow_R t_2 \ \Rightarrow \ (\exists t')(t_1 \stackrel{*}{\Rightarrow}_R t' \ \wedge \ t_2 \stackrel{*}{\Rightarrow}_R t'))$$

Lemma 1 (Newman Lemma)
Let R be a terminating trs. Then, R is confluent iff it is locally confluent.

Theorem 7 Confluence of finite and terminating trs is decidable.