REWRITE SYSTEMS

Nachum Dershowitz and Jean-Pierre Jouannaud

Chapter 6 of
Handbook of Theoretical Computer Science
Volume B: Formal Methods and Semantics
J. van Leeuwen, ed.
pages 243–320
North-Holland
Amsterdam, 1990

1 INTRODUCTION

Equations are ubiquitous in mathematics and the sciences. Sometimes one tries to determine if an identity follows logically from given axioms; other times, one looks for solutions to a given equation. These reasoning abilities are also important in many computer applications, including symbolic algebraic computation, automated theorem proving, program specification and verification, and high-level programming languages and environments.

Rewrite systems are directed equations used to compute by repeatedly replacing subterms of a given formula with equal terms until the simplest form possible is obtained. The idea of simplifying expressions has been around as long as algebra has. As a form of computer program, rewrite systems made their debut in [Gorn, 1967]; many modern programs for symbolic manipulation continue to use rewrite rules for simplification in an *ad hoc* manner. As a formalism, rewrite systems have the full power of Turing machines and may be thought of as non-deterministic Markov algorithms over terms, rather than strings. (Regarding Markov algorithms, see, e.g. [Tourlakis, 1984].) The theory of rewriting is in essence a theory of normal forms; to some extent it is an outgrowth of the study of Church's Lambda Calculus and Curry's Combinatory Logic.

To introduce some of the central ideas in rewriting, we consider several variations on the "Coffee Can Problem" (attributed to Carel Scholten in [Gries, 1981]). Imagine a can containing coffee beans of two varieties, *white* and *black*, arranged in some order. Representing the contents of the can as a sequence of bean colors, e.g.

$$white\ white\ black\ black\ white\ white\ black\ black,$$

the rules of our first game are as follows:

$$\begin{array}{rcl} black\ white & \rightarrow & black \\ white\ black & \rightarrow & black \\ black\ black & \rightarrow & white \end{array}$$

This set of rules is an example of a rewrite system. Each rule describes a legal move: the first two state that, at any stage of the game, the white bean of any adjacent pair of different beans may be discarded; the last rule states that two adjacent black beans may be replaced by one white one (an unlimited supply of white beans is on hand). For example, the following is a possible sequence of moves (the underlined beans participate in the current move):

$$\begin{array}{c} white\ white\ black\ \underline{black\ white}\ white\ black\ black \\ white\ white\ \underline{black\ black}\ white\ black\ black \\ white\ white\ white\ \underline{white\ black}\ black \\ white\ white\ \underline{white\ black}\ black \\ white\ \underline{white\ black}\ black \\ \underline{white\ black}\ black \\ \underline{black\ black} \\ white \end{array}$$

The object of this game is to end up with as few beans as possible. It is not hard to see that with an odd number of black beans the game will always end with one black bean, since the "parity" of black beans is unchanging. Played right (always keeping at least one black bean around), an even (non-zero) number of black beans leads to one white bean, but other pure-white results are also possible. For instance, applying the third rule right off to both pairs of black beans leaves six white beans.

By adding an additional rule, the above game may be modified to always end in a single bean:

$$\begin{array}{rcl} black\ white & \rightarrow & black \\ white\ black & \rightarrow & black \\ black\ black & \rightarrow & white \\ white\ white & \rightarrow & white \end{array}$$

What is different about this new game is that one of the rules applies to any can containing more than one bean. What is interesting is that the outcome of the game is completely independent of the choice of which move is made when. To establish this, an analysis of the different possible divergences helps. The order in which moves at non-overlapping locations are made is immaterial, since whichever moves were not taken can still be taken later. The critical cases occur when the possible moves overlap and making one precludes making the other. For example, from *white black black*, either *black black* or *white white* can result. The point is that these two situations can both lead to the same single-white state. The same is true for other overlapping divergences. With this independence in mind, it is a trivial matter to predict the deterministic outcome of any game, by picking a sequence amenable to a simple analysis. Indeed, any initial state with an even number of black beans must now end in one white one. (The "semantic" argument given in [Gries, 1981], based on the invarying parity of black beans, requires some insight, whereas the above analysis is entirely mechanical, as we will see later in this chapter.)

It is obvious that neither of the above games can go on forever, since the number of beans is reduced with each move. A potentially much longer, but still finite, game is:

$$\begin{array}{rcl} \textit{black white} & \rightarrow & \textit{white white white black} \\ \textit{white black} & \rightarrow & \textit{black} \\ \textit{black black} & \rightarrow & \textit{white white white white} \\ \textit{white white} & \rightarrow & \textit{white} \end{array}$$

The new rules have the same end-effect as the original ones: no matter how often a bean-increasing move is made, in the final analysis the can must be emptied down to one bean.

Finally, we consider a variant (Gries' original problem) in which the rules apply to *any* two (not necessarily adjacent) beans. The new rules are

$$\begin{array}{rcl} \textit{black} \ \ldots \ \textit{white} & \rightarrow & \textit{black} \ \ldots \\ \textit{white} \ \ldots \ \textit{black} & \rightarrow & \textit{black} \ \ldots \\ \textit{black} \ \ldots \ \textit{black} & \rightarrow & \textit{white} \ \ldots \\ \textit{white} \ \ldots \ \textit{white} & \rightarrow & \textit{white} \ \ldots \end{array}$$

where the ellipsis on the right refers to the same beans as covered by its counterpart on the left. The new rules, in effect, allow a player to "shake" the can prior to making a move. Again, it can be shown that the outcome is uniquely determined by the initial setup and is, consequently, the same as that of the previous two games.

The final result of an unextendible sequence of rule applications is called a "normal form." Rewrite systems defining at most one normal form for any input term can serve as functional programs or as interpreters for equational programs [O'Donnell, 1977a]. When computations for equal terms always terminate in a unique normal form, a rewrite system may be used as a non-deterministic functional program [Goguen-Tardo, 1979]. Such a system also serves as a procedure for deciding whether two terms are equal in the equational theory defined by the rules, and, in particular, solves the "word problem" for that theory. Knuth [Knuth-Bendix, 1970] devised an effective test (based on critical overlaps) to determine for any given terminating system if, in fact, all computations converge to a canonical form, regardless of the non-deterministic choices made. In that seminal paper, it was also demonstrated how failure of the test (as transpires for the first Coffee Can game) often suggests additional rules that can be used to "complete" a non-convergent system into a convergent one. The discovery [Fay, 1979] that convergent rewrite systems can also be used to enumerate answers to satisfiability questions for equational theories led to their application [Dershowitz, 1984] within the logic programming paradigm.

Rewriting methods have turned out to be among the more successful approaches to equational theorem proving. In this context, completion is used for "forward reasoning," while rewriting is a form of "backward reasoning." Completion utilizes an ordering on terms to provide strong guidance during forward reasoning and to direct the simplification of equations. Besides the use of convergent systems as decision procedures, Lankford [1975] proposed that completion-like methods supplant paramodulation for equational deduction within resolution-based theorem provers; later, Hsiang [1982] showed how a variant of completion can be used in place of resolution for (refutational) theorem proving in first-order predicate calculus. Although completion often generates an infinite number of additional rules, and—at the same time—deletes many old rules, Huet [1981] demonstrated that "fairly" implemented completion serves as a semi-decision procedure for the equational theory defined by the given equations when it does not abort (something it might be forced to do on account of equations that cannot be directed without loss of termination). Lankford's procedure paramodulates to circumvent failure of completion. Rewriting techniques have also been applied [Musser, 1980] to proving inductive theorems by showing that no contradiction can result from assuming the validity of the theorem in question.

In the next two sections, we take a brief look at the syntax and semantics of equations from the algebraic, logical, and operational points of view. To use a rewrite system as a decision procedure, it must be convergent; this fundamental concept is studied in Section 4 as an abstract property of binary relations. To use a rewrite system for computation or as a decision procedure for validity of identities, the termination property is crucial; basic methods for proving termination are presented in Section 5. Section 6 is devoted to the question of satisfiability of equations. Then, in Section 7, we return to the convergence property as applied to rewriting. The completion procedure, its extensions, refinements, and main uses, are examined in Section 8. Brief mention of variations on the rewriting theme is made in the final section.

1.1 Further Reading

Previous surveys of term rewriting include [Huet-Oppen, 1980; Buchberger-Loos, 1982; Jouannaud-Lescanne, 1987; Klop, 1987].

2 SYNTAX

Algebraic data types are an important application area for rewrite-based equational reasoning. In the abstract approach to data specification, data are treated as abstract objects and the semantics of functions operating on data are described by a set of constraints. When constraints are given in the form of equations, a specification is called *algebraic* [Guttag, 1976]. In this section, we talk about the syntax of equations and of equational proofs. As we will see, by turning equations into left-to-right "rules", a useful concept of "direct" proof is obtained.

2.1 Terms

Suppose we wish to define the standard stack operations, *top* and *pop*, as well as an operation *alternate* that combines two stacks. Stacks of natural numbers can be represented by terms of the form $push(s_1, push(s_2, \ldots, push(s_n, \Lambda)\ldots))$, where Λ is the empty stack and the s_i denote representations of natural numbers, 0, $succ(0)$, $succ(succ(0))$, and so on. The precise syntax of these representations can be given in the following way:

$$\begin{array}{rcl} Zero & = & \{0\} \\ Nat & = & Zero \cup succ(Nat) \\ Empty & = & \{\Lambda\} \\ Stack & = & Empty \cup push(Nat, Stack) \end{array}$$

The left sides of these equations name sets of different kinds of terms. An expression like $succ(Nat)$ denotes the set of all terms $succ(s)$, with $s \in Nat$. The symbols 0, *succ*, Λ, and *push*, used to build data, are called "constructors;" any term built according to these rules is a *constructor term*.

To specify the desired stack operations, we must also define the syntax of non-constructor terms:

$$\begin{array}{rcrcl} top & : & Stack & \rightarrow & Nat \\ pop & : & Stack & \rightarrow & Stack \\ alternate & : & Stack \times Stack & \rightarrow & Stack \end{array}$$

Then we give semantics to the new functions by constraining them to satisfy the following set of equations:

$$\begin{array}{rcl} top(push(x,y)) & = & x \\ pop(push(x,y)) & = & y \\ alternate(\Lambda, z) & = & z \\ alternate(push(x,y),z) & = & push(x, alternate(z,y)) \end{array}$$

(where x, y, and z are variables ranging over all data of the appropriate type). Inverses of constructors, like *top* and *pop*, are called *selectors*. With these equations, it can be shown, for example, that

$$alternate(push(top(push(0,\Lambda)),\Lambda), pop(push(succ(0),\Lambda))) \quad = \quad push(0,\Lambda).$$

A specification is said to be "sufficiently complete" if, according to the semantics, every term is equal to a term built only from constructors; the above operations are not well-defined in this sense, since terms like $pop(\Lambda)$ are not equal to any constructor term. (See Section 3.2.)

In general, given a (denumerable) set $\mathcal{F} = \cup_{n\geq 0}\mathcal{F}_n$ of function symbols—called a (finitary) *vocabulary* or *signature*—and a (denumerable) set $\mathcal{X}$ of variable symbols, the set of (first-order) *terms* $\mathcal{T}(\mathcal{F},\mathcal{X})$ over $\mathcal{F}$ and $\mathcal{X}$ is the smallest set containing $\mathcal{X}$ such that $f(t_1,\ldots,t_n)$ is in $\mathcal{T}(\mathcal{F},\mathcal{X})$ whenever $f \in \mathcal{F}_n$ and $t_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ for $i = 1,\ldots,n$. The stack example uses $\mathcal{F}_0 = \{0,\Lambda\}$, $\mathcal{F}_1 = \{top, pop, succ\}$, $\mathcal{F}_2 = \{push, alternate\}$, and $\mathcal{X} = \{x,y,z\}$. The syntax of the stack example also differentiates between terms of type Nat and of type $Stack$. Categorizing function symbols, variables, and terms into classes, called *sorts*, can be very helpful in practice; from now on, however, we will suppose that there is only one, all-inclusive sort. All concepts developed here can be carried over to the many-sorted case, as will be sketched in the Section 9.

Each symbol f in $\mathcal{F}$ has an *arity* (*rank*) which is the index n of the set $\mathcal{F}_n$ to which it belongs. (We will assume that the $\mathcal{F}_n$ are disjoint, though "varyadic" vocabularies pose little problem.) In a well-formed term, each symbol of arity n has n immediate subterms. Elements of arity zero are called *constants*, of which we will always make the (sometimes critical) assumption that there is at least one constant. Terms in $\mathcal{T}(\mathcal{F}_0 \cup \mathcal{F}_1, \mathcal{X})$ are called *monadic*; they are "words" spelled with unary symbols (from $\mathcal{F}_1$) and ending in a constant (from $\mathcal{F}_0$) or variable (from $\mathcal{X}$). Variable-free terms are called *ground* (or *closed*); the set $\mathcal{T}(\mathcal{F},\emptyset)$ of ground terms will be denoted by $\mathcal{G}(\mathcal{F})$. Note that $\mathcal{G}(\mathcal{F})$ is non-empty by the previous assumption. We will often use $\mathcal{T}$ to refer to a set of terms $\mathcal{T}(\mathcal{F},\mathcal{X})$, with $\mathcal{F}$ and $\mathcal{X}$ left unspecified, and $\mathcal{G}$ to refer to the corresponding set of ground terms. Occasionally, we use prefix or postfix notation for $\mathcal{F}_1$ and infix for $\mathcal{F}_2$.

A term t in $\mathcal{T}(\mathcal{F},\mathcal{X})$ may be viewed as a finite ordered tree, the leaves of which are labeled with variables (from $\mathcal{X}$) or constants (from $\mathcal{F}_0$) and the internal nodes of which are labeled with function symbols (from $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots$) of positive arity, with outdegree equal to the arity of the label. A *position* within a term may be represented—in Dewey decimal notation—as a sequence of positive integers, describing the path from the outermost, "root" symbol to the head of the subterm at that position. By $t|_p$, we denote the *subterm* of t rooted at position p. For example, if $t = push(0, pop(push(y,z)))$, then $t|_{2.1}$ is the first subterm of t's second subterm, which is $push(y,z)$. Positions are often called *occurrences*; we will use this latter denomination to refer, instead, to the subterm $t|_p$. We write $t \trianglerighteq s$ to mean that s is a subterm of t and also write $t[s]$ to indicate that s occurs within t. We speak of position p as being *above* position q in some term t if p (represented as a sequence of numbers) is a prefix of q, i.e. if occurrence $t|_q$ is within $t|_p$. A subterm of t is called *proper* if it is distinct from t.

Reasoning with equations requires replacing subterms by other terms. The term t with its subterm $t|_p$ replaced by a term s is denoted by $t[s]_p$. We refer to any term u that is the same as t everywhere except below p, i.e. such that $u[s]_p = t$, as the *context* within which the replacement takes place; more precisely, a context is a term u with a distinguished position p.

A *substitution* is a special kind of replacement operation, uniquely defined by a mapping from variables to terms, and written out as $\{x_1 \mapsto s_1, \ldots, x_m \mapsto s_m\}$ when there are only finitely many variables x_i not mapped to themselves. Formally, a substitution σ is a function from $\mathcal{X}$ to $\mathcal{T}(\mathcal{F},\mathcal{X})$, extended to a function from $\mathcal{T}$ to itself (also denoted σ and for which we use postfix notation) in such a way that $f(t_1,\ldots,t_n)\sigma = f(t_1\sigma,\ldots,t_n\sigma)$, for each f (of arity n) in $\mathcal{F}$ and for all terms $t_i \in \mathcal{T}$. A term t *matches* a term s if $s\sigma = t$ for some substitution σ; in that case we write $s \trianglelefteq t$ and also say that t is an *instance* of s or that s *subsumes* t. The relation $\trianglelefteq$ is a quasi-ordering on terms, called *subsumption*.[1] For example, $f(z) \trianglelefteq f(a)$, and $f(z) \trianglelefteq f(f(a))$, since z is "less specific" than a or $f(a)$. On the other hand, $f(x)$ and $f(z)$ are equally general; we write $f(x) \doteq f(z)$, where $\doteq$ is the equivalence relation associated with $\trianglelefteq$, called *literal similarity* (α-*conversion* in λ-calculus parlance; *renaming*, in other circles). Subsumption and the subterm ordering are special cases of the *encompassment* quasi-ordering, in which $s \trianglerighteq t$ if a subterm of t is an instance of s. (Encompassment is called "containment" in [Huet, 1981].) For example, $f(z) \trianglerighteq g(f(a))$, since $f(a)$ is an instance of $f(z)$.

The *composition* of two substitutions, denoted by juxtaposition, is just the composition of the two functions; thus, if $x\sigma = s$ for some variable x, then $x\sigma\tau = s\tau$. We say that substitution σ is *at least as general* as substitution ρ, or that ρ is an *instance* of σ if there exists a substitution τ such that $\sigma\tau = \rho$; we use the same symbols to denote this quasi-ordering on substitutions, as we used for subsumption of terms. For example, $\{x \mapsto a, y \mapsto f(a)\} \trianglerighteq \{x \mapsto y, y \mapsto f(z)\} \doteq \{x \mapsto z, y \mapsto f(x)\}$. Here, and everywhere, we use the mirror image of a binary relation symbol like $\trianglelefteq$ for its inverse.

In this survey, we will mainly be dealing with binary relations on terms that possess the following fundamental properties:

Definition 1. A binary relation $\rightarrow$ over a set of terms $\mathcal{T}$ is a *rewrite relation* if it is closed both under context application (the "replacement" or "monotonicity" property) and under substitutions (the "fully invariant property"). A transitive and irreflexive rewrite relation will be called a *rewrite ordering*.

In other words, $\rightarrow$ is a rewrite relation if $s \rightarrow t$ implies $u[s\sigma]_p \rightarrow u[t\sigma]_p$, for all terms s and t in $\mathcal{T}$, contexts u, positions p, and substitutions σ. The inverse, symmetric closure, reflexive closure, and transitive closure of any rewrite relation are also rewrite relations.

To fix nomenclature, the letters a through h will be used for function symbols; l, r, and s through w will denote arbitrary terms; x, y, and z will be reserved for variables; p and q, for positions; lower case Greek letters, for substitutions. Binary relations will frequently be denoted by arrows of one kind or another. If $\rightarrow$ is a binary relation, then $\leftarrow$ is its inverse, $\leftrightarrow$ is its symmetric closure ($\leftarrow \cup \rightarrow$), $\rightarrow^=$ is its reflexive closure ($\rightarrow \cup =$), $\rightarrow^*$ is its reflexive-transitive closure ($\rightarrow \circ \cdots \circ \rightarrow$) and $\rightarrow^+$ is its transitive closure ($\rightarrow \circ \rightarrow^*$).

2.2 Equations

Replacement leads to the important notion of "congruence:" an equivalence relation $\approx$ on a set of terms is a *congruence* if $f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)$ whenever $s_i \approx t_i$ for $i = 1,\ldots,n$. In particular, the reflexive-symmetric-transitive closure $\leftrightarrow^*$ of any rewrite relation $\rightarrow$ is a congruence. Note that rewrite relations and congruences form a complete lattice with respect to intersection.

Our primary interest is in congruences generated by instances of equations. For our purposes, an *equation* is an unordered pair $\{s,t\}$ of terms. (For other purposes, it is preferable to regard equations as ordered pairs.) Equations will be written in the form $s = t$.[2] The two terms may contain variables; these are understood as being universally quantified. Given a (finite or infinite) set of equations E over a set of terms $\mathcal{T}$, the *equational theory* of E, $\mathcal{T}h(E)$, is the set of equations that can be obtained by taking reflexivity, symmetry, transitivity, and context application (or functional reflexivity) as inference rules and all instances of equations in E as axioms. Thus, if E is recursively-enumerable, so are its theorems $\mathcal{T}h(E)$. We write $E \vdash s = t$ if $s = t \in \mathcal{T}h(E)$.

---

[1] A *quasi-ordering* $\precsim$ is any reflexive and transitive binary relation; the associated equivalence relation $\approx$ is the intersection of $\precsim$ with its inverse; the associated "strict" (i.e. irreflexive) partial order $\prec$ is their difference.

[2] To avoid confusion, authors are frequently forced to use a different symbol in the syntax of equations, instead of the heavily overloaded "equals sign"—a precaution we choose not to take in this survey.

A more compact inference system is based on the familiar notion of "replacement of equals for equals" (a.k.a. Leibniz's Law). We write $s \leftrightarrow_E t$, for terms s and t in $\mathcal{T}$, if s has a subterm that is an instance of one side of an equation in E and t is the result of replacing that subterm with the corresponding instance of the other side of the equation. Formally, $s \leftrightarrow_E t$ if $s = u[l\sigma]_p$ and $t = u[r\sigma]_p$ for some context u, position p in u, equation $l = r$ (or $r = l$) in E, and substitution σ. It is folk knowledge that $E \vdash s = t$ iff $s \leftrightarrow_E^* t$, where $\leftrightarrow_E^*$ is the reflexive-transitive closure of $\leftrightarrow_E$; in other words, two terms are provably equal if one may be obtained from the other by a finite number of replacements of equal subterms. The relation $\leftrightarrow_E$ is the "rewrite" closure of E, when the latter is viewed as a symmetric relation, and $\leftrightarrow_E^*$ is the congruence closure of $\leftrightarrow_E$, i.e. $\leftrightarrow_E^*$ is the smallest congruence over $\mathcal{T}$ such that $l\sigma \leftrightarrow_E^* r\sigma$ for all equations $l = r$ in E and substitutions σ over $\mathcal{T}$. We will write $[\![s]\!]_E$ for the congruence class of a term s, and denote by $\mathcal{T}/E$ the set of all congruence classes, i.e. the quotient of the set $\mathcal{T}(\mathcal{F}, \mathcal{X})$ of terms and the *provability* relation $\leftrightarrow_E^*$.

A *derivation* in E is any sequence $s_0 \leftrightarrow_E s_1 \leftrightarrow_E \cdots \leftrightarrow_E s_i \leftrightarrow_E \cdots$ of applications of equational axioms in E. A *proof* in E of an equation $s = t$ is a "justified" finite derivation $s = s_0 \leftrightarrow_E \cdots \leftrightarrow_E sn = t$ $(n \geq 0)$, each step $s_i \leftrightarrow_E s_{i+1}$ of which is justified by reference to an axiom $l = r$ in E, a position p_i in s_i, and a substitution σ_i, such that $s_i|_{p_i} = l\sigma_i$ and $s_{i+1} = s_i[r\sigma_i]_{p_i}$. Returning to our stack specification, and letting E be its axioms, the following is an example of a derivation:

$$\begin{array}{c} alternate(push(top(push(0, z)), z), \Lambda) \\ \leftrightarrow_E \quad alternate(push(0, z), \Lambda) \quad \leftrightarrow_E \\ alternate(push(0, pop(push(succ(y), z))), \Lambda) \end{array}$$

The first step may be justified by the axiom $top(push(x, y)) = x$, position 1.1, and substitution $\{x \mapsto 0, y \mapsto z\}$; the second step, by the axiom $pop(push(x, y)) = y$ (used from right to left), position 1.2, and substitution $\{x \mapsto succ(y), y \mapsto z\}$.

2.3 Rewrite Rules

The central idea of rewriting is to impose directionality on the use of equations in proofs. Unlike equations which are unordered, a *rule* over a set of terms $\mathcal{T}$ is an ordered pair $\langle l, r\rangle$ of terms, which we write as $l \rightarrow r$. Rules differ from equations by their use. Like equations, rules are used to replace instances of l by corresponding instances of r; unlike equations, rules are not used to replace instances of the right-hand side r. A (finite or infinite) set of rules R over $\mathcal{T}$ is called a *rewrite system*, or (more specifically) a *term-rewriting system*. A system R may be thought of as a (non-symmetric) binary relation on $\mathcal{T}$; the rewrite closure $\rightarrow_R$ of this relation describes the effect of a left-to-right application of a rule in R.

Definition 2. For given rewrite system R, a term s in $\mathcal{T}$ *rewrites* to a term t in $\mathcal{T}$, written $s \rightarrow_R t$, if $s|_p = l\sigma$ and $t = s[r\sigma]_p$, for some rule $l \rightarrow r$ in R, position p in s, and substitution σ.

This is the same as saying that $s = u[l\sigma]_p$ and $t = u[r\sigma]_p$, for some context u and position p in u. A subterm $s|_p$ at which a rewrite can take place is called a *redex*; we say that s is *irreducible*, or in *normal form*, if it has no redex, i.e. if there is no t in $\mathcal{T}$ such that $s \rightarrow_R t$.

Systems of rules are used to compute by rewriting repeatedly, until, perhaps, a normal form is reached. A *derivation* in R is any (finite or infinite) sequence $t_0 \rightarrow_R t_1 \rightarrow_R \cdots \rightarrow_R t_i \rightarrow_R \cdots$ of applications of rewrite rules in R. The *reducibility*, or *derivability*, relation is the quasi-ordering $\rightarrow_R^*$, i.e. the reflexive-transitive closure of $\rightarrow_R$. We write $s \rightarrow_R^! t$ if $s \rightarrow_R^* t$ and t is irreducible, in which case we say that t is a normal form *of* s. This *normalizability* relation $\rightarrow_R^!$ is not a rewrite relation, since normalizing a subterm does not mean that its superterm is in normal form. One says that a rewrite system is *normalizing* if every term has at least one normal form.

A *ground rewriting-system* is one all the rules of which are ground (i.e. elements of $\mathcal{G} \times \mathcal{G}$); an important early paper on ground rewriting is [Rosen, 1973]. A *string-rewriting system*, or *semi-Thue system*, is one that has monadic words ending in the same variable (i.e. strings of elements of $\mathcal{T}(\mathcal{F}_1, \{x\})$) as left- and right-hand side terms; [Book, 1987] is a survey of string rewriting. The (first three) Coffee Can Games can be formulated as string-rewriting systems, with *white* and *black* as monadic symbols. A *left-linear* system is one in which no variable occurs more than once on any left-hand side. (Ground- and string-rewriting systems are special cases of left-linear systems, with no variable and one variable per term, respectively.)

For our purposes, one of the most essential properties a rewrite system R can enjoy is *unique normalization*, by which is meant that every term t in $\mathcal{T}$ possesses exactly one normal form. The normalizability relation $\rightarrow^!_R$ for uniquely-normalizing systems defines a function, and we denote by $R(t)$ the value of that function for a term t in $\mathcal{T}$. If all sequences of rewrites lead to a unique normal form, the system will be called *convergent*. A rewrite system R is said to be *(inter-)reduced* if, for each rule $l \rightarrow r$ in R, the right-hand side r is irreducible under R and no term s less than l in the encompassment ordering $\trianglerighteq$ is reducible. (For convergent R, this is equivalent to the standard definition [Huet, 1981] which requires that the left-hand side l not be rewritable by any other rule.) As we will see this, too, is a convenient property. We will reserve the adjective *canonical* for reduced convergent systems, though, in the literature, "canonical" is usually synonymous with "convergent".

Orienting the equations of the stack example gives a canonical rewrite system R:

$$\begin{array}{rcl} top(push(x,y)) & \rightarrow & x \\ pop(push(x,y)) & \rightarrow & y \\ alternate(\Lambda, z) & \rightarrow & z \\ alternate(push(x,y),z) & \rightarrow & push(x, alternate(z,y)) \end{array}$$

(That every term has at least one normal form will be shown in Section 5.3; that there can be no more than one normal form will be shown in Section 7.2.) A *derivation* is any (finite or infinite) chain $t_1 \rightarrow_R t_2 \rightarrow_R \cdots$ of rewrite steps. An example is:

$$\begin{array}{l} alternate(push(top(push(0,z)),z),\Lambda) \quad \rightarrow_R \quad alternate(push(0,z),\Lambda) \\ \qquad \rightarrow_R \quad push(0, alternate(\Lambda,z)) \quad \rightarrow_R \quad push(0,z). \end{array}$$

The first step is an application of the *top-push* rule at the occurrence $top(push(0,z))$; the second is an application of the *alternate-push* rule, with 0 for x, z for y, and Λ for z; the third, of $alternate(\Lambda,z) \rightarrow z$. Note that an alternative derivation is possible, leading to the same normal form:

$$\begin{array}{l} alternate(push(top(push(0,z)),z),\Lambda) \quad \rightarrow_R \quad push(top(push(0,z)), alternate(\Lambda,z)) \\ \qquad \rightarrow_R \quad push(top(push(0,z)),z) \quad \rightarrow_R \quad push(0,z). \end{array}$$

Operationally, then, rewriting is a non-deterministic computation, with the choice of rule and position left open. For convergent systems, the choice among possible rewrites at each step does not affect the normal form computed for any given input term.

Definition 3. A binary relation $\rightarrow$ on a set T is *terminating* if there exists no endless chain $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \cdots$ of elements of T.

(Such relations have sometimes been called *Noetherian* in the term-rewriting literature—after the algebraicist, Emily Noether—though the adjective is ordinarily used to exclude infinite ascending chains.) Termination is more than being normalizing, since the latter allows some derivations to be infinite. A partial (irreflexive) ordering $\succ$ of a set T is *well-founded* if there exists no infinite descending chain $t_1 \succ t_2 \succ \cdots$ of elements of T. Thus, a relation $\rightarrow$ is terminating iff its transitive closure $\rightarrow^+$ is a well-founded ordering. The importance of terminating relations lies in the possibility of inductive proofs in which the hypothesis is assumed to hold for all elements t such that $s \rightarrow^+ t$ when proving it for arbitrary s. Induction on terminating relations, sometimes called "Noetherian induction," is essentially well-founded induction (i.e. transfinite induction extended to partial orderings); see, for example, [Cohn, 1981]. We will have occasion to employ this technique in Sections 4.1 and 8.2.

A rewrite system R is *terminating* for a set of terms $\mathcal{T}$ if the rewrite relation $\rightarrow_R$ over $\mathcal{T}$ is terminating, i.e. if there are no infinite derivations $t_1 \rightarrow_R t_2 \rightarrow_R \cdots$ of terms in $\mathcal{T}$. When a system is terminating,

Figure 1: Joinability properties of relations.

every term has at least one normal form. Note that a terminating system cannot have any rule, like $alternate(y, \Lambda) \rightarrow pop(push(x, y))$, with a variable on the right that is not also on the left (since x could, for example, be $top(alternate(y, \Lambda))$), nor can a left-hand side be just a variable, like $z \rightarrow alternate(\Lambda, z)$. These two restrictions are often placed *a priori* on rewrite rules (cf. [Huet, 1980]), something we prefer not to do. Methods for establishing termination are described in Section 5.

A *valley*, or *rewrite*, proof for a system R of an equation $s = t$ takes the form $s \rightarrow_R^* v \leftarrow_R^* t$, in which the same term v is reached by rewriting s and t. Here is an example, using the above system:

$$\begin{array}{ccc} alternate(push(top(push(0,z)),z),\Lambda) & & alternate(push(0,pop(push(y,z))),\Lambda) \\ \rightarrow_R \quad alternate(push(0,z),\Lambda) & & push(0,alternate(\Lambda,pop(push(y,z)))) \quad \leftarrow_R \\ & \rightarrow_R \quad push(0,alternate(\Lambda,z)) \quad \leftarrow_R & \end{array}$$

With a terminating system of only a finite number of rules, the search space for rewrite proofs is finite. Of course, there is—in general—no guarantee that such a "direct" proof exists for a particular consequence of the equations represented by R. When that is the case, i.e. when the relation $\leftrightarrow_R^*$ is contained in $\rightarrow_R^* \circ \leftarrow_R^*$, the system is called *Church-Rosser*, after a property in [Church-Rosser, 1936]. See Figure 1(a). Equivalent properties are defined in Section 4 and methods of establishing them are described in Section 7.

2.4 Decision Procedures

One of our main concerns is in decision procedures for equational theories; an early example of such a procedure for groups is [Dehn, 1911]. Terminating, Church-Rosser rewrite systems are convergent and define unique normal forms. For a convergent system R to determine provability in its underlying equational theory (treating its rules as equational axioms), it should have only a finite number of rules. Then, to decide if $s = R$ t, one can test if computing the normal forms $R(s)$ and $R(t)$ results in the same term. The Church-Rosser property means that $\leftrightarrow_R^* = \rightarrow_R^* \circ \leftarrow_R^*$; termination ensures that $\rightarrow_R^* \circ \leftarrow_R^* = \rightarrow_R^! \circ \leftarrow_R^!$; finiteness of R makes $\rightarrow_R^!$ decidable; and Church-Rosser implies that $\rightarrow_R^!$ defines a function $R(\cdot)$. Thus, a system R provides a decision procedure for the equational theory of a set of axioms E if R is (a) finite, (b) terminating, (c) Church-Rosser, and (d) sound and adequate for E. Here, *soundness* means that the rules of R are contained in the relation $\leftrightarrow_E^*$, and *adequacy* means that E is contained in $\leftrightarrow_R^*$; together, they imply that $\leftrightarrow_E^* = \leftrightarrow_R^*$.

In our stack example, R is sound and adequate for E; hence, R decides equality in E. In general, a system R with the properties (a-d) is said to be *complete* for E.

Given a set of equations E, its *word problem* is the question whether an arbitrary equation $s = t$ between two *ground* terms follows from E. The word problem is, thus, a special case of provability in E for arbitrary equations. If R is a convergent system for E, its word problem is decidable by reducing s and t to their R-normal form. Actually, it is enough if R is *ground-convergent*, that is, if every ground term has a unique normal form. Many rewriting-system decision procedures are known; perhaps the first rewriting-based decision procedure for a word problem is the one in [Evans, 1951] for "loops". When a rewriting decision procedure exists, it can be very effective. In Section 8, we will elaborate on systematic methods used to generate convergent systems from given equational axioms.

Of course, not all equational theories can be decided by rewriting—for a variety of reasons. First, some theories (classes of equations) are not *finitely based*; for such theories there exists no finite set of axioms from which all other equations in the theory follow. An example [Taylor, 1979] is the intersection of the theories (consequences) of the following two semigroups (abbreviating products by juxtaposition and exponentiation):

$$\begin{array}{|rcl|}\hline x(yz) & = & (xy)z \\ (xyz)^2 & = & x^2y^2z^2 \\ x^3y^3z_1^2z_2^3 & = & y^3x^3z_1^2z_2^3 \\ \hline\end{array} \qquad \begin{array}{|rcl|}\hline x(yz) & = & (xy)z \\ x^3y^3 & = & y^3x^3 \\ & & \\ \hline\end{array}$$

Nor are all finitely-based equational theories decidable, the first counter-examples having been given by Markov [1947] and Post [1947], in a slightly different context. (See the interesting historical comments at the end of [Tarski-Givant, 1985].) A prime example of an undecidable equational theory is Combinatory Logic (with binary infix symbol "$\cdot$" and constants S, K, and I):

$$\begin{array}{rcl} I \cdot x & = & x \\ (K \cdot x) \cdot y & = & x \\ ((S \cdot x) \cdot y) \cdot z & = & (x \cdot z) \cdot (y \cdot z) \end{array}$$

(see [Curry-Feys, 1958]). Most disconcertingly, there are finitely-based, decidable theories for which there can be no rewriting-system decision procedure. For example, no finite system—even over an enlarged vocabulary—can rewrite any two terms, equal by commutativity, to the same term [Dershowitz-etal, 1988]. In other words, no term-rewriting system can decide validity in the decidable theory defined by the commutativity axiom, $x \cdot y = y \cdot x$, since that equation, oriented in either direction, gives a non-terminating rewrite system. Deciding word problems is a somewhat different question, since then one looks only at ground equations over a finite vocabulary. But this same example (let $\mathcal{F} = \{0, succ, \cdot\}$) demonstrates that not all decidable word problems can be decided by a finite rewrite system over the same vocabulary; cf. [Klop, 1987]. The following is one of the simple (semigroup) theories with an undecidable word problem given in [Matijasevic, 1967] ($\mathcal{F}_0 = \{a, b\}$, $\mathcal{F}_2 = \{\cdot\}$, and products are abbreviated as before):

$$\begin{array}{rcl} x(yz) & = & (xy)z \\ aba^2b^2 & = & b^2a^2ba \\ a^2bab^2a & = & b^2a^3ba \\ aba^3b^2 & = & ab^2aba^2 \\ b^3a^2b^2a^2ba & = & b^3a^2b^2a^4 \\ a^4b^2a^2ba & = & b^2a^4 \end{array}$$

2.5 Extensions

Happily, rewriting techniques can be adapted to handle some of the more important cases, in which any orientation of the axioms yields a non-terminating system. The simplest example of a "structural" axiom requiring special treatment is commutativity: any system containing either $x \cdot y \to y \cdot x$ or $y \cdot x \to x \cdot y$ is perforce non-terminating. We describe two techniques for dealing with such axioms, "class rewriting" and "ordered rewriting."

A *(congruence-) class-rewriting system* comes in two parts: rules and equations. By R/S we denote the class system composed of a set $R = \{l_i \rightarrow r_i\}$ of rewrite rules and a set $S = \{u_i \leftrightarrow v_i\}$ of equations, the latter written with double-headed arrows to stress their symmetrical usage. Generalizing the notion of term rewriting, we say that s rewrites to t *modulo* S, denoted $s \rightarrow_{R/S} t$, if $s \leftrightarrow_S^* u[l\sigma]_p$ and $u[r\sigma]_p \leftrightarrow_S^* t$, for some context u, position p in u, rule $l \rightarrow r$ in R, and substitution σ. Thus, R is essentially computing in the quotient set $\mathcal{T}/S = \{[\![t]\!]_S | t \in \mathcal{T}\}$ of S-congruence classes (more precisely, $\leftrightarrow_S^*$-congruence classes), rewriting a term by rewriting any S-equivalent term. Class-rewriting systems were introduced in [Lankford-Ballantyne, 1977a] for *permutative* congruences, that is, congruences for which each congruence class is finite. Of great practical importance are associative-commutative (AC) rewrite systems, where S is an equational system consisting of associativity and commutativity axioms for a subset of the binary symbols (in $\mathcal{F}_2$). The last Coffee Can Game can be formulated as an AC system, by using an associative-commutative operator for adjacency with the four rules of the second game. Then, *black white black* can rewrite directly to *black black* or, via *black black white*, to *white white*.

The notions of derivation and normal form extend naturally to class-rewriting systems. We say that R/S is terminating if $\rightarrow_{R/S}$ is terminating and that it is *Church-Rosser modulo* S if $\leftrightarrow_{R/S}^*$ is contained in $\rightarrow_{R/S}^* \circ \leftrightarrow_S^* \circ \leftarrow_{R/S}^*$. More generally, any rewrite relation $\rightarrow_T$ is Church-Rosser modulo S if $\leftrightarrow_{S\cup T}^* \subseteq \rightarrow_T^* \circ \leftrightarrow_S^* \circ \leftarrow_T^*$; see Figure 1(f). For instance, let BA/AC be the following class system (over vocabulary $\mathcal{F}_2 = \{and, xor\}$ and $\mathcal{F}_0 = \{F, T\}$):

BA			*AC*		
$and(x,T)$	$\rightarrow$	x			
$and(x,F)$	$\rightarrow$	F	$and(x,y)$	$\leftrightarrow$	$and(y,x)$
$and(x,x)$	$\rightarrow$	x	$and(x, and(y,z))$	$\leftrightarrow$	$and(and(x,y),z)$
$xor(x,F)$	$\rightarrow$	x	$xor(x,y)$	$\leftrightarrow$	$xor(y,x)$
$xor(x,x)$	$\rightarrow$	F	$xor(x, xor(y,z))$	$\leftrightarrow$	$xor(xor(x,y),z)$
$and(xor(x,y),z)$	$\rightarrow$	$xor(and(x,z), and(y,z))$			

This system is convergent, i.e. terminating and Church-Rosser modulo AC [Hsiang, 1982]; it computes normal forms of Boolean-ring expressions that are unique up to permutations under associativity and commutativity. (The exclusive-or normal form is due to [Zhegalkin, 1927; Stone, 1936].)

The idea, then, is to put equations that cannot be handled by rewriting into S, placing in R only rules that preserve termination. If R/S is also Church-Rosser modulo S, then $s \leftrightarrow_{R\cup S}^* t$ iff their normal forms are S-equivalent. For this to work, there are two additional considerations: S-equivalence must be decidable, and R/S normal forms must be effectively computable. The latter requirement does not, however, come automatically, even if R is finite and S-equivalence is decidable, since a rule in R is applicable to a term when any S-equivalent term contains an instance of a left-hand side, whereas S-equivalence classes need be neither finite nor computable. Note that a class system R/S cannot be terminating if (R is non-empty and) S contains an equation with a variable on one side not also on the other, or if it contains an axiom like idempotency, $x \cdot x = x$, with a lone variable on one side and more than one occurrence of it on the other.

Even if S-equivalence classes are computable, they may be impractically large, making class-rewriting prohibitively expensive. These difficulties with R/S are usually circumvented by using a weaker rewrite relation, introduced in [Peterson-Stickel, 1981]. We denote this relation by $\rightarrow_{S\backslash R}$ (others have used $\longrightarrow_{R,S}$); under it, a term is rewritten only if it has a subterm that is equivalent to an instance of a left-hand side. We call $\rightarrow_{S\backslash R}$ the "extended" rewrite relation for R; our notation is meant to suggest that S-steps are not applied above the R-step. Formally:

Definition 4. For given rewrite system R and congruence relation S, the S-*extended* rewrite relation $\rightarrow_{S\backslash R}$ is defined by $s \rightarrow_{S\backslash R} t$, for two terms s, t in $\mathcal{T}$, iff $s|_p \leftrightarrow_S^* l\sigma$ and $t = s[r\sigma]_p$, for some rule $l \rightarrow r$ in R, position

p in s, and substitution σ.

The notions of normal-form, etc., are analogous to the previous definitions. We say that $S \backslash R$ is Church-Rosser modulo S if if $\rightarrow_{S \backslash R}$ is, i.e. any two terms, equal in $R \cup S$, lead to S-equivalent terms via extended rewriting with $S \backslash R$.

S-extended rewriting avoids the need to compute S-congruence classes, requiring instead an S-matching algorithm: We say that a term t *S-matches* l, if there exists a substitution σ such that $l\sigma \leftrightarrow_S^* t$. Matching algorithms are known for many theories, including associativity, commutativity, and associativity with commutativity; see Section 6.2. If R is finite and S-matching substitutions are computable, then $\rightarrow_{S \backslash R}$ is computable, too.

The relation $\rightarrow_{S \backslash R}$ is a subset of $\rightarrow_{R/S}$, and hence does not necessarily render the same normal forms. For the above system BA/AC, we have $and(a, and(a, b)) \longrightarrow_{BA/AC} and(a, b)$, but $and(a, and(a, b)) \not\longrightarrow_{BA \backslash AC} and(a, b)$. However, it is often the case that by adding certain consequences as new rules, the two relations can be made to coincide, as shown in [Peterson-Stickel, 1981] for AC, and in [Jouannaud-Kirchner, 1986] for the general case. When R/S is terminating and $S \backslash R$ is Church-Rosser modulo S, the theory $R \cup S$ can be decided by computing $S \backslash R$-normal forms and testing for S-equivalence. For example, if BA is augmented with the two rules, $and(x, and(x, y)) \rightarrow and(x, y)$ and $xor(x, xor(x, y)) \rightarrow y$, then AC-extended rewriting suffices to compute the normal forms of BA/AC. In Section 4.2, conditions for equivalence of normal forms are discussed.

An *ordered-rewriting systems* also comes in two parts: a set of equations and an ordering. Ordered rewriting does not require that a particular equation always be used from left-to-right. Instead, an equation may be used in whichever direction agrees with the given ordering on terms. Suppose, for example, that $x \cdot y = y \cdot x$ is an equation and that $a \cdot b$ is greater than $b \cdot a$ in the ordering. Then, we would use commutativity to rewrite $a \cdot b$ to the normal form $b \cdot a$, but not vice-versa.

Definition 5. Given a set E of equations over a set of terms $\mathcal{T}$ and a rewrite ordering (transitive and irreflexive rewrite relation) $\succ$ over $\mathcal{T}$, a term s in $\mathcal{T}$ *rewrites* to a term t in $\mathcal{T}$ *according* to $\succ$, denoted $s \rightarrow_{E\succ} t$, or just $s \rightarrow_{\succ} t$, if $s = u[l\sigma]_p$, $t = u[l\sigma]_p$ and $l\sigma \succ r\sigma$, for some context u, position p in u, equation $l = r$ in E, and substitution σ.

This corresponds to considering each instance $l\sigma = r\sigma$ of an equation as a rewrite rule going one way or the other. Thus, the ordered-rewriting relation $\rightarrow_{\succ}$, is contained in the rewrite relation obtained from the intersection of the two rewrite relations, $\leftrightarrow_E$ and $\succ$; in particular, $s \succ t$ if $s \rightarrow_{\succ} t$. The relation $\rightarrow_{\succ}$ is equivalent to the intersection when $\succ$ is total.

2.6 Further Reading

The standard work on the Lambda Calculus is [Barendregt, 1984]; its role in the semantics of functional programming is discussed in [Barendregt, 1989]. Evans [1951] and Knuth [Knuth-Bendix, 1970] pioneered the use of rewrite systems as decision procedures for validity in equational theories. Bledsoe [1977] was an early advocate of incorporating rewriting techniques in general-purpose theorem provers.

3 SEMANTICS

As is usual in logic, models give meaning to syntactic constructs. In our case, the models of equational theories are just algebras, that is, sets with operations on their elements, and provability by equational reasoning coincides with truth in all algebras. When a rewrite system has the unique normalization (convergence) property, a term's normal form can be its "meaning". It turns out that the set of irreducible terms of a convergent system yields an algebra that is "free" among all the algebras satisfying the axioms expressed by its rules. The free algebra is that model in which the only equalities are those that are valid in all models.

3.1 Algebras

Let $\mathcal{F} = \cup_n \mathcal{F}_n$ be a signature. An $\mathcal{F}$-*algebra* $\mathbf{A}$ consists of a non-empty domain of values, called the *universe* (or *carrier*, or *underlying set*), which we also denote A (when feasible, we will use boldface for the algebra and italics for the corresponding universe), and a family $\mathcal{F}_{\mathbf{A}}$ of $\mathcal{F}$-indexed *(finitary) fundamental operations*, such that for every symbol f of arity n in $\mathcal{F}_n$, the corresponding operation $f_{\mathbf{A}}$ in $\mathcal{F}_{\mathbf{A}}$ maps A^n to A. Since we presume the existence of at least one constant (in $\mathcal{F}_0$), universes will always be non-empty. Given an assignment $\theta : \mathcal{X} \rightarrow A$ of values to each of the variables in $\mathcal{X}$, the $\mathcal{F}$-algebra $\mathbf{A}$ attaches a meaning to each term t of $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which is the result of applying the operations corresponding to each function symbol in t using the values assigned by θ for the variables. Let E be a set of equations; an algebra $\mathbf{A}$ is a *model* of E if, for every equation $s = t$ in E, and for every assignment of values to variables in s and t, the meanings of s and t are identical. (From now on, we generally omit reference to $\mathcal{F}$ and just speak of "algebras".) By $\mathcal{M}od(E)$ we denote the class of all models of E, each of which is an algebra.

Consider the algebra $\mathbf{A}$ whose universe contains two elements, a black coffee bean and a white one: $A = \{black, white\}$. The algebra has two operations: a unary operation "invert" which turns a white bean into a black one and vice-versa, and a binary operation "move" which takes two beans and returns a bean according to the rules of the second Coffee Can Game of Section 1. It is easy to see that this algebra is a model of the associative axiom $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, interpreting "$\cdot$" as "move," since the result of a game does not depend upon the order of moves. However, interpreting the unary symbol "$^-$" as "invert" and the identity constant 1 as *white* does not yield a group, since a *black white* move gives a black bean, and not the identity element. To obtain a group (as might be preferred by some mathematicians), we must change the rules of the game slightly:

$$\begin{array}{rcl} black\ white & \rightarrow & black \\ white\ black & \rightarrow & black \\ black\ black & \rightarrow & white \\ white\ white & \rightarrow & white \end{array}$$

This algebra is a model of all three group axioms.

A class $\mathcal{K}$ of algebras is a *variety* if there exists a set E of equations such that $\mathcal{K} = \mathcal{M}od(E)$. For example, though groups are axiomatizable non-equationally by giving one associative operator "$\cdot$" and a constant 1 satisfying $\forall x \exists y (x \cdot y = 1)$, they may also be axiomatized in the following way:

$$\begin{array}{rcl} 1 \cdot x & = & x \\ x^- \cdot x & = & 1 \\ (x \cdot y) \cdot z & = & x \cdot (y \cdot z) \end{array}$$

Groups are actually "one-based," with the following axiom providing a basis:

$$x/(((x/x)/y)/z)/(((x/x)/x)/z)) \quad = \quad y$$

[Higman-Neumann, 1952]. Note that groups defined in the latter way give a different variety than the previous axiomatization, since their signatures differ; nevertheless, the two equational theories are essentially the same, since the operations of one are definable in terms of the other (in particular, $x/y = x \cdot y^-$ and $x \cdot y = x/((x/x)/y)$). Rings, commutative rings, and lattices are also varieties; fields are not. Tarski has endeavored to equationally axiomatize the foundations of mathematics; see [Tarski-Givant, 1985]. Huet [1985] has shown that much of category theory is equational.

A mapping ϕ is a *homomorphism* from algebra $\mathbf{A}$ to algebra $\mathbf{B}$, if $f_{\mathbf{A}}(a_1, \ldots, a_n)\phi = f_{\mathbf{B}}(a_1\phi, \ldots, a_n\phi)$, for all $f \in \mathcal{F}_n$ and $a_i \in A$. An *isomorphism* is a bijective homomorphism. Any assignment $\sigma : \mathcal{X} \rightarrow B$ of values to variables extends in this way to a homomorphism $\sigma : \mathcal{T} \rightarrow \mathbf{B}$ by letting $f(t_1, \ldots, t_n)\sigma = f_{\mathbf{B}}(t_1\sigma, \ldots, t_n\sigma)$. An equation $s = t$ is *valid* (or *true*) in a specific algebra $\mathbf{B}$ if, for all assignments σ of values in B to variables in s and t, $s\sigma$ and $t\sigma$ represent the same element of B. "Satisfiability" is the dual of validity: an equation is *satisfiable* in an algebra if it has a solution in that algebra, that is, if there is an assignment of values to variables for which both sides yield the same value. Validity of an equation $s = t$ (that is, validity in *all* models) is expressed as $\mathcal{M}od(E) \models s = t$, or $s =_E t$ for short.

Varieties are characterized in the following algebraic way [Birkhoff, 1935]: A class of algebras $\mathcal{K}$ is a variety iff it is closed under Cartesian products, subalgebras, and homomorphic images. That is, a class $\mathcal{K}$ of algebras is a variety if (a) for any $\mathbf{A}_1, \ldots, \mathbf{A}_n$ in $\mathcal{K}$ ($n \geq 0$), their product $\mathbf{A}_1 \times \cdots \times \mathbf{A}_n$ is also in $\mathcal{K}$, where $f_{\mathbf{A}_1 \times \cdots \times \mathbf{A}_n}(\ldots\langle a_1, \ldots, a_n\rangle \ldots) = \langle f_{\mathbf{A}_1}(\ldots a_1 \ldots), \ldots, f_{\mathbf{A}_n}(\ldots a_n \ldots)\rangle$; (b) for any subset B of A for algebra $\mathbf{A}$ in $\mathcal{K}$, the subalgebra obtained by restricting $f_{\mathbf{A}}$ to B for each f in $\mathcal{F}$ is also in $\mathcal{K}$; and (c) for any homomorphism $\theta : A \to B$ between universes, if A is in $\mathcal{K}$, then so is the algebra $\mathbf{B}$ wherein $f_{\mathbf{B}}(\ldots a_i\theta \ldots) = f_{\mathbf{A}}(\ldots a_i \ldots)\theta$. This result of Birkhoff's can be used to show that an operation is not equationally axiomatizable. For instance, the models of strict[3] $if \cdot then \cdot else \cdot$ are not closed under products, hence, no set of equations can characterize that operation. Still, it is remarkable that equational axioms E can be given for $if \cdot then \cdot else \cdot$ such that an equation is valid for $\mathcal{M}od(E)$ iff it is valid in the "if-then-else" models [Bloom-Tindell, 1983; Guessarian-Meseguer, 1987].

Let E be a set of equations. Clearly, replacement of equals for equals is sound, i.e. $E \vdash s = t$ implies $\mathcal{M}od(E) \models s = t$ for all s and t. For the other direction, consider the quotient algebra $\mathcal{T}/E$ (described in Section 2.2). It is one of the models of E. Since classes $\mathcal{T}/E$ are defined by the congruence $\leftrightarrow_E^*$, which is just replacement of equals, we have $\mathcal{T}/E \models s = t$ implies $E \vdash s = t$. Together, we get the following:

Completeness Theorem (Birkhoff-1935). *For any set of equations E and terms s and t in $\mathcal{T}$, $\mathcal{M}od(E) \models s = t$ iff $\mathcal{T}/E \models s = t$ iff $E \vdash s = t$.*

Accordingly, we may use the semantic notion $=_E$ and syntactic notion $\leftrightarrow_E^*$ interchangeably. It follows that a convergent rewrite system R decides validity for the models of its rules, since $s \leftrightarrow_R^* t$ iff $R(s) = R(t)$.

A substitution is a homomorphism from $\mathcal{T}$ to itself. If there exists a substitution $\sigma : \mathcal{T} \to \mathcal{T}$ such that $s\sigma$ and $t\sigma$ are identical, then for any algebra $\mathbf{B}$ there exists a homomorphism $\theta : \mathcal{T} \to \mathbf{B}$ such that $s\theta = t\theta$. In other words, if an equation is satisfiable in the term algebra $\mathcal{T}$, then it is satisfiable in all algebras. Similarly, any equation satisfiable in the quotient algebra $\mathcal{T}/E$ is satisfiable in all algebras in $\mathcal{M}od(E)$. Satisfiability in $\mathcal{T}$ is called *unifiability*; satisfiability in $\mathcal{T}/E$ is called *E-unifiability*. The unification problem is the subject of Section 6.

3.2 Initial Algebras

For many purposes, not all models are of equal interest. One generally asks whether an equation is valid in a specific model. For example, one might ask whether the equation $alternate(y, \Lambda) = y$ is true for all stacks y. (Of course, all equations are valid, let alone satisfiable, in a trivial algebra having only one element in its domain.) For applications like abstract data types, attention is often focused on those "standard" models that are (finitely) generated from the signature itself, in which every element of the domain is the interpretation of some term.

An algebra $\mathbf{A}$ in a class $\mathcal{K}$ of algebras is *free* over a set $\mathcal{X}$ of variables if $\mathcal{X}$ is a subset of A and, for any algebra $\mathbf{B} \in \mathcal{K}$ and assignment $\theta : \mathcal{X} \to B$, there exists a unique homomorphism $\phi : \mathbf{A} \to \mathbf{B}$ such that ϕ and θ agree on $\mathcal{X}$. A free algebra is unique up to isomorphism, whenever it exists. The free algebra over $\mathcal{X}$ among all algebras is just (isomorphic to) the *term algebra* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ with the the symbol $f \in \mathcal{F}$ itself as the operator $f_{\mathcal{T}}$. An algebra $\mathbf{A}$ in a class $\mathcal{K}$ of algebras is *initial* if, for any algebra $\mathbf{B}$ in $\mathcal{K}$ there exists a unique homomorphism $\phi : \mathbf{A} \to \mathbf{B}$. The initial object among all $\mathcal{F}$-algebras is (isomorphic to) the *ground-term algebra* $\mathcal{G}(\mathcal{F})$, again with the function symbol itself as operator, and corresponds to the Herbrand universe over the symbols in $\mathcal{F}$. The importance of the initial algebra lies in its uniqueness (it is the free algebra for empty $\mathcal{X}$), and in the fact that the class $\mathcal{K}$ consists of its homomorphic images, making it the most "abstract" among them.

Among all models of a set of equations E, the prototypical one is the initial algebra $I(E)$ of E. Its universe consists of one element for each E-congruence class of ground terms. In other words, $I(E)$ is (isomorphic

[3] An operation is *strict*, or *naturally extended*, if it yields the undefined value whenever one of its arguments is undefined.

to) the quotient $\mathcal{G}/E$ of the ground-term algebra $\mathcal{G}$ and the congruence $\leftrightarrow_E^*$ (restricted to $\mathcal{G}$). This algebra can be realized if R is a ground-convergent rewrite system for E, since, then, E-equivalent ground terms have the same R-normal form. Accordingly, the *normal-form algebra* of R has the set of ground R-normal forms as its universe and operations f_R defined by $f_R(t_1, \ldots, t_n) = R(f(t_1, \ldots, t_n))$ for all normal forms t_i. This algebra is (isomorphic to) the initial algebra $I(R)$ of the variety defined by the rules in R (considered as equations) [Goguen, 1980]. Thus, rewriting computes ground normal forms that are representatives of their congruence classes. It is in this sense that rewriting is a "correct" implementation of initial-algebra semantics. Specification languages based on abstract data types, such as OBJ [Futatsugi-etal, 1985], follow this implementation scheme: equations are used as rewrite rules, and unique normalization is needed for the operational and initial algebra semantics to coincide.

Exactly those variable-free equations that follow necessarily from E hold in the initial algebra. Thus, the word problem for E, i.e. deciding, for *ground* terms s and t, whether $s = t$ holds in every model of E, is the same as determining if $I(E) \models s = t$. More generally, one may ask if an equation $s = t$ (possibly containing variables) is valid in the initial algebra $I(E)$, which is the case iff all of its ground instances hold for $\mathcal{M}od(E)$. We will write $s =_{I(E)} t$ as an abbreviation for $I(E) \models s = t$ and call the class $\mathcal{I}nd(E)$ of equations $s = t$ valid in $I(E)$ the *inductive theory* of E. It is easy to verify that the relation $=_{I(E)}$ is a congruence over $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Unlike equational theories, inductive theories are not necessarily recursively enumerable (even for finite E). The inductive theory includes all the equations in the equational theory; on the other hand, an equation that holds in the initial model need not hold in all models, i.e. the inclusion $\mathcal{T}h(E) \subseteq \mathcal{I}nd(E)$ may be strict. Tarski [1968] dubbed *ω-complete* those equational theories that coincide with the associated inductive theory, but ω-completeness is not possible, in general [Henkin, 1977]. Finally, note that if an equation $s = t$ is not valid in $I(E)$, that means that some ground equation $u = v$ which does not hold for $\mathcal{M}od(E)$ does hold for $\mathcal{M}od(E \cup \{s = t\})$.

For example, let $\mathcal{F} = \{0, succ, \Lambda, push, alternate\}$ and let E be

$$\begin{array}{rcl} alternate(\Lambda, z) & = & z \\ alternate(push(x, y), z) & = & push(x, alternate(z, y)) \end{array}$$

The equation $alternate(y, \Lambda) = y$ is valid in $I(E)$, since it is provable for all ground terms of the form $push(s_1, push(s_2, \ldots, push(s_n, \Lambda) \ldots))$, and all other ground terms (entailing *alternate*) are provably equal to one of this form. It is not, however, valid in a model $\mathbf{A}$ that, besides the usual stacks, includes stacks built on top of another empty-stack value, Λ_0, and for which $alternate_{\mathbf{A}}(\Lambda_0, \Lambda_{\mathbf{A}}) = \Lambda_{\mathbf{A}}$. Thus, by Birkhoff's Completeness Theorem, $alternate(y, \Lambda) = y$ is not an equational consequence of the given axioms.

For a system R to correctly implement an algebraic specification E, it is enough that their inductive theories are the same, i.e. that $\mathcal{I}nd(E) = \mathcal{I}nd(R)$, when the rules of R are considered as equations. For example, the convergent three-rule system

$$\begin{array}{rcl} alternate(\Lambda, z) & \rightarrow & z \\ alternate(y, \Lambda) & \rightarrow & y \\ alternate(push(x, y), z) & \rightarrow & push(x, alternate(z, y)) \end{array}$$

is a correct implementation of the above specification E of *alternate*, since all equations in E are deductive theorems of R and all rules in R are inductive theorems of E.

The notion of sufficient completeness of function definitions (and its relation to software specification) was introduced in [Guttag, 1976]:

Definition 6. Let the set $\mathcal{F}$ of function symbols be split into a set $\mathcal{C}$ of *constructors* and a set $\mathcal{F} - \mathcal{C}$ of other symbols. Let E be a set of equations in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The specification E is *sufficiently complete* (or "has no junk") with respect to $\mathcal{C}$, if every ground term t in $\mathcal{G}(\mathcal{F})$ is provably equal to a constructor term s in $\mathcal{G}(\mathcal{C})$.

Definition 7. Let $\mathcal{C}$ be a set of constructors and let E be a set of equations split into a set $E_{\mathcal{C}}$ of equations in $\mathcal{T}(\mathcal{C}, \mathcal{X})$ and a set $E_{\mathcal{F}-\mathcal{C}}$ of other equations. The constructors $\mathcal{C}$ are said to be *free* when $E_{\mathcal{C}}$ is empty. The specification E is *consistent* (or "has no confusion") with respect to $\mathcal{C}$, if for arbitrary ground constructor terms s and t in $\mathcal{G}(\mathcal{C})$, $s =_E t$ iff $s =_{E_\mathcal{C}} t$.

This generalizes the standard notion of consistency, which is with respect to the Boolean values T and F. For example, the previous specification for stacks becomes inconsistent with respect to the free constructors $\{0, succ, \Lambda, push\}$, if it is "enriched" with the equation $alternate(y, \Lambda) = alternate(y, y)$, since that implies $push(0, \Lambda) = push(0, push(0, \Lambda))$. Adding an equation $push(x, push(y, z)) = push(y, push(x, z))$ to $E_\mathcal{C}$ makes the constructors non-free and constructor terms represent unordered "bags." When a set of equations is both consistent and sufficiently complete, it is reasonable to consider it a "specification" of the functions in $\mathcal{F} - \mathcal{C}$. In this case, the algebra $\mathcal{G}(\mathcal{F})/E$, considered as a $\mathcal{C}$-algebra, is isomorphic to $\mathcal{G}(\mathcal{C})/E_\mathcal{C}$. This allows one to build complex specifications from simpler ones. Unfortunately, both properties are undecidable in general (see [Guttag, 1976]).

As mentioned above, term-rewriting is used to compute in the initial algebra. More generally, if R/S is a ground-convergent class-rewriting (or ordered-rewriting) system, then the normal-form algebra, with universe $R(\mathcal{G}) = \{[\![R(t)]\!]_S | t \in \mathcal{G}\}$ and operations defined by $f_R(a_1, \ldots, a_n) = [\![R(f(a_1, \ldots, a_n))]\!]_S$ for all $a_i \in R(\mathcal{G})$, is initial for the variety defined by $R \cup S$, and $R(\mathcal{G})$ is isomorphic to $I(R \cup S)$. In implementing a sufficiently complete specification, one would want all ground normal forms to be constructor terms, i.e. that $R(\mathcal{G}(\mathcal{F})) \subseteq \mathcal{G}(\mathcal{C})$.

In certain cases, sufficient completeness can be related to the following more tractable property:

Definition 8. For any rewrite relation $\rightarrow_T$, a term s in $\mathcal{T}$ is *ground T-reducible*, if all its ground instances $s\gamma \in \mathcal{G}$ are rewritable by $\rightarrow_T$.

Suppose that (a) a ground-convergent class-rewriting system R/S is complete for an equational specification E, (b) the R/S-normal form of any ground constructor term in $\mathcal{G}(\mathcal{C})$ is a ground constructor term, and (c) S does not equate any ground constructor term with a ground non-constructor term in $\mathcal{G}(\mathcal{F}) - \mathcal{G}(\mathcal{C})$. For a system R/S satisfying these properties, E is sufficiently complete iff all terms $f(x_1, \ldots, x_n)$ are ground R/S-reducible, when f is in $\mathcal{F} - \mathcal{C}$ and x_i are distinct variables in $\mathcal{X}$. The rationale is that, by ground reducibility, any non-constructor term t must contain a reducible subterm, and, since the system is terminating and sound for E, t must be equal to a constructor term. This connection between sufficient completeness and ground reducibility is implicit in [Plaisted, 1985]. If each left-hand side of a rule in R and each side of an equation in S contains a non-constructor symbol, then property (b) is ensured; cf. [Huet-Hullot, 1980].

Ground reducibility is decidable for finite R and empty S [Plaisted, 1985; Kapur-etal, 1987]. A faster decision method is obtained by reducing ground reducibility to the emptiness problem of the language produced by a "conditional tree grammar" describing the system's ground normal forms [Comon, 1989]. Testing for ground R-reducibility, however, requires exponential time, even for left-linear R [Kapur-etal, 1987]. In the special case where all constructors are free, ground reducibility is more easily testable. This case had been considered in [Nipkow-Weikum, 1982] for left-linear systems. The general case was considered in [Dershowitz, 1985] and [Kounalis, 1985]. The former defines a "test set" for ground-reducibility by instantiating $f(x_1, \ldots, x_n)$ in all possible ways up to a bound that depends on the maximal depth of a left-hand side; the latter constructs a smaller test set, computed by repeated unification of $f(x_1, \ldots, x_n)$ with left-hand sides, and improves on [Thiel, 1984]. Ground R/S-reducibility is undecidable when S is a set of associative-commutative axioms [Kapur-etal, 1987], but is decidable when R is left-linear [Jouannaud-Kounalis, 1989].

For ground-convergent systems R, any equation between distinct R-normal forms is considered to be inconsistent with R (considering all symbols in $\mathcal{F}$ as constructors). The observation that an equation $s = t$ is valid in the initial algebra $I(R)$ iff no inconsistency follows from $R \cup \{s = t\}$ is the basis of the *proof by consistency* method of inductive theorem proving (for proving theorems in $\mathcal{I}nd(R)$), pioneered by Musser [1980] (and so named in [Kapur-Musser, 1987]). If there exists a ground-convergent system R', with the same ground normal forms as R, and which presents the same equational theory as $R \cup \{s = t\}$, then inconsistency is precluded [Lankford, 1981]. It can readily be shown that $R(\mathcal{G}) \subseteq R'(\mathcal{G})$, for any two systems R and R', iff every left-hand side of R' is ground R-reducible [Dershowitz, 1982b;

Jouannaud-Kounalis, 1989]. It follows that R and R' have the same inductive theory if they are both ground convergent and every left-hand side of one system is ground reducible by the other. This method, relating validity in the initial algebra to ground-reducibility, extends to class-rewriting, with ground R/S-reducibility replacing its ordinary counterpart [Goguen, 1980; Lankford, 1981; Jouannaud-Kounalis, 1989]. In Section 8, we will consider how to search for an appropriate R'.

Note that the equation $alternate(y, \Lambda) = y$ is *not* an inductive theorem of the earlier stack specification, given at the beginning of Section 2.1, even though it holds for all stacks Λ, $push(s_1, \Lambda)$, etc. The problem is that the full vocabulary has a richer set of ground terms, involving *top* and *pop*, but their specification is not sufficiently complete. In particular, the equation in question does not hold true for "error" terms like $pop(\Lambda)$: $alternate(pop(\Lambda), \Lambda) = pop(\Lambda)$ does not follow from the axioms. Test-set based methods for proofs in the (non-initial) constructor model are described in [Kapur-etal, 1986; Zhang, 1988]; see also [Kapur-Musser, 1987].

3.3 Computable Algebras

When a system R is not terminating, rewriting will not necessarily compute a representative for the congruence class of a term. However, as long as R is Church-Rosser, one knows that *if* a normal form is obtained, it is unique. Of course, one can always turn a finite set of equations into a Church-Rosser system by turning each equation into a symmetric pair of rules, but then no term at all has a normal form. More interesting is the ability to code interpreters for functional languages as Church-Rosser systems that are normalizing for input programs that terminate for the given input values; see [O'Donnell, 1977b]. Furthermore, there are computational strategies (that is, specific choices of where to rewrite next), such as not forever ignoring an "outermost" redex (one that is not a subterm of another redex), that are guaranteed to result in a normal form whenever there is one [O'Donnell, 1977a]. An optimal strategy, i.e. one with normalizing derivations of minimal length, is not in general computable [Huet-Levy, 1990].

Turing machine computations can be simulated by rewrite systems in at least two different ways: by systems of monadic rules that rewrite instantaneous descriptions according to the machine's transitions [Huet-Lankford, 1978], and by a (non-monadic) one-rule system in which the transitions appear as part of the terms ([Dauchet-1989], refining [Dershowitz, 1987]). Thus, rewrite systems provide a fully general programming paradigm (to the extent that Church's Thesis defines "fully general"). These constructions also imply that most interesting properties, including convergence, are in general undecidable. On the other hand, equality (the word problem) is decidable in what are called "computable" algebras [Meseguer-Goguen, 1985]; see [Wirsing, 1989].

3.4 Further Reading

For a survey of equational logic, see [Taylor, 1979]. A comprehensive multi-volume work on varieties is [McKenzie-etal, 1987; Freese-etal, 1989]. Some relevant recent results are summarized in [McNulty, 1989]. A detailed exposition of algebraic aspects of rewriting is [Meseguer-Goguen, 1985]; algebraic semantics are the subject of [Wirsing, 1989].

4 CHURCH-ROSSER PROPERTIES

Newman [1942] developed a general theory of "sets of moves," that is, of arbitrary binary relations. It has since become customary to deal separately with properties of such abstract binary relations and with those of relations on terms. In our discussion of the Church-Rosser property, we continue in that tradition, putting off almost all mention of rewrite systems to later sections.

4.1 Confluence

In Section 2, we defined the Church-Rosser property for rewrite systems. The analogous property can hold for any binary relation:

Definition 9. A binary relation $\rightarrow$ on any set T is *Church-Rosser* if its reflexive-symmetric-transitive closure $\leftrightarrow^*$ is contained in the *joinability* relation $\rightarrow^* \circ \leftarrow^*$.

See Figure 1(a). This is equivalent to the following simpler property:

Definition 10. A binary relation $\rightarrow$ on any set T is *confluent* if the relation $\leftarrow^* \circ \rightarrow^*$ is contained in the joinability relation $\rightarrow^* \circ \leftarrow^*$.

Confluence says that no matter how one diverges from a common ancestor, there are paths joining at a common descendent. Sometimes the notation $\uparrow$ is used for the common ancestor relation and $\downarrow$ for joinability (common descendent); then confluence boils down to $\uparrow \subseteq \downarrow$. See Figure 1(b). The equivalence with the Church-Rosser property [Newman, 1942], can be shown by a simple inductive argument on the number of divergences $\leftarrow^* \circ \rightarrow^*$ making up $\leftrightarrow^*$.

For arbitrary $\rightarrow$, define $s \rightarrow^! t$ iff $s \rightarrow^* t$ and there is no u such that $t \rightarrow u$ and call t the *normal form* of s. Confluence implies the impossibility of more than one normal form. A binary relation $\rightarrow$ on a set is *strongly confluent* if any local divergence $\leftarrow \circ \rightarrow$ is contained in the immediate descendent relation $\rightarrow^= \circ \leftarrow^=$, i.e. if for any *peak* $s \leftarrow u \rightarrow t$ of elements s, t, and u, one of the following four cases holds: $s = t$, $s \leftarrow t$, $s \rightarrow t$, or $s \rightarrow v \leftarrow t$ (for some element v). See Figure 1(c). (A slightly weaker definition is given in [Huet, 1980], namely $\leftarrow \circ \rightarrow \subseteq \rightarrow^* \circ \leftarrow^=$, which also allows for circumstances like $a \rightarrow b$, $a \rightarrow c$, $b \rightarrow d \rightarrow c$, and $c \rightarrow d \rightarrow b$.) Strong confluence implies confluence [Newman, 1942] by a "tiling" argument. Strong confluence is used in the classical proofs of the Church-Rosser property for the λ-calculus, since confluence of $\rightarrow$ is exactly strong confluence of $\rightarrow^*$ (see [Barendregt, 1984]).

Definition 11. A binary relation $\rightarrow$ on any set T is *locally confluent* if any local divergence $\leftarrow \circ \rightarrow$ is contained in the joinability relation $\rightarrow^* \circ \leftarrow^*$.

See Figure 1(d). Local confluence does not generally imply confluence; see the counter-examples in Figure 2, due to [Newman, 1942] and [Hindley, 1964].

However:

Diamond Lemma ([Newman, 1942]). *A terminating relation is confluent iff it is locally confluent.*

The name derives from the pictorial proof in Figure 3, due to [Huet, 1980], which uses induction with respect to the terminating relation. When $\rightarrow$ is terminating, it follows from the above results that it is Church-Rosser iff $\leftarrow \circ \rightarrow$ is contained in $\rightarrow^! \circ \leftarrow^!$.

Confluence is sometimes established by well-founded induction in the following way: Let $\succ$ be a well-founded ordering on the elements and suppose that for every "peak" $s \leftarrow u \rightarrow t$ there exists an (undirected) path $s = w_0 \leftrightarrow w_1 \leftrightarrow \cdots \leftrightarrow w_n = t$ $(n \geq 0)$ such that $u \succ w_1, \ldots, w_{n-1}$. Then it can be shown (by induction on multisets of elements; see Section 5.1) that $\rightarrow$ is Church-Rosser [Winkler-Buchberger, 1983]. See Figure 1(e).

4.2 Coherence

As preparation for the study, in the Section 7.3, of Church-Rosser properties of extended-rewriting with a congruence, we consider here abstract properties of combinations of an arbitrary binary relation $\rightarrow_R$ and a symmetric binary relation $\leftrightarrow_S$, both on the same set. Let $\rightarrow_T$ be a relation lying anywhere between $\rightarrow_R$ and the quotient relation $\rightarrow_{R/S}$. Note that $\leftrightarrow^*_{R \cup S} = \leftrightarrow^*_{S \cup T}$. In what follows, we will also use R, S, T, and R/S to refer to the relations $\rightarrow_R$, $\leftrightarrow_S$, $\rightarrow_T$, and $\rightarrow_{R/S}$, respectively. At one extreme, T can be R, a case partially dealt with in [Sethi, 1974; Huet, 1980]; at the other extreme, T can be R/S; the general case—considered here—was first studied in [Jouannaud-Kirchner, 1986].

Figure 2: Two locally-confluent relations.

Figure 3: Proof of Diamond Lemma.

If one is going to compute normal forms with T instead of with R, the natural question to ask is if their normal forms are equivalent. That is, under what conditions is $\leftarrow^!_R \circ \rightarrow^!_T$ contained in $\leftrightarrow^*_S$? The relation T is Church-Rosser modulo S if $\leftrightarrow^*_{S\cup T}$ is contained in $\rightarrow^*_T \circ \leftrightarrow^*_S \circ \leftarrow^*_T$; see Figure 1(f). When T is terminating and Church-Rosser modulo S, one can determine if $s \leftrightarrow^*_{R\cup S} t$, by finding T-normal forms of s and t and testing them for S-equivalence. That is, $\leftrightarrow^*_{R\cup S}$ is equivalent to $\leftrightarrow^*_{S\cup T}$ which is contained in $\rightarrow^!_T \circ \leftrightarrow^*_S \circ \leftarrow^!_T$; in particular, R/S- and T-normal forms must be equivalent.

When R/S is terminating, the Church-Rosser property may be decomposed into two local ones: T is *locally coherent modulo S with R* if $\leftarrow_T \circ \rightarrow_R$ is contained in $\rightarrow^*_T \circ \leftrightarrow^*_S \circ \leftarrow^*_T$; T is *locally coherent modulo S with S* if $\leftarrow_T \circ \leftrightarrow_S$ is contained in $\rightarrow^*_T \circ \leftrightarrow^*_S \circ \leftarrow^*_T$. Compare Figures 2(g) and 2(h) with 2(d). The concept of coherence was developed by [Jouannaud-Kirchner, 1986] and generalizes *compatibility*, as in [Peterson-Stickel, 1981].

Coherence Lemma ([Jouannaud-Kirchner, 1986]). *Let $R \subseteq T \subseteq R/S$. If R/S is terminating, then T is Church-Rosser modulo S iff it is locally coherent modulo S with both R and S.*

One proves that $\leftrightarrow^*_{R\cup S}$ is contained in $\rightarrow^!_T \circ \leftrightarrow^*_S \circ \leftarrow^!_T$ by induction on the multiset of elements in a path $t_0 \leftrightarrow_{R\cup S} t_1 \leftrightarrow_{R\cup S} \cdots \leftrightarrow_{R\cup S} t_n$. The induction is with respect to the well-founded ordering on multisets (see Section 5.1) induced by $\rightarrow^+_{R/S}$. It then follows that $\leftrightarrow^*_{S\cup T}$ is contained in $\rightarrow^*_T \circ \leftrightarrow^*_S \circ \leftarrow^*_T$.

4.3 Further Reading

Newman's 1942 paper [Newman, 1942] defined the basic notions. Huet [1980] introduced the use of (Noetherian) induction on terminating relations for studying these notions. Confluence (the vanilla-flavored kind) and many related properties of relations are discussed in [Klop, 1987].

5 TERMINATION

Recall that a rewrite system R is terminating for a set of terms $\mathcal{T}$ if there are no infinite derivations $t_1 \rightarrow_R t_2 \rightarrow_R \cdots$ of terms in $\mathcal{T}$. The following is one example of a nonterminating system [Toyama, 1987b]:

$$\begin{array}{rcl} f(a,b,x) & \rightarrow & f(x,x,x) \\ g(x,y) & \rightarrow & x \\ g(x,y) & \rightarrow & y \end{array}$$

The depth (i.e. the maximum nesting of symbols) of a term in any of its derivations is bounded by the depth of the initial term, but it has a cycling derivation starting from $f(g(a,b), g(a,b), g(a,b))$. If $\rightarrow_R$ is contained in some well-founded partial ordering $\succ$ on $\mathcal{T}$, then R is obviously terminating. The rule $f(f(x)) \rightarrow f(g(f(x)))$, for instance, is terminating, since the number of adjacent f's is reduced with each application. In general, it is undecidable whether a system is terminating, even if both sides of all rules are monadic [Huet-Lankford, 1978] or if it has only one left-linear rule [Dauchet, 1989]. For ground systems, however, termination is decidable [Huet-Lankford, 1978]. The decidability of termination of non-length-increasing string-rewriting systems is open.

5.1 Reduction Orderings

The above method of establishing termination requires one to reason about the global effect of applying a rule at a subterm. To avoid consideration of the infinite number of possible contexts, one can use well-founded orderings on terms:

Definition 12. A *reduction ordering* on a set of terms $\mathcal{T}$ is any well-founded rewrite ordering of $\mathcal{T}$.

Termination is assured if each of the *rules* in R is contained in a reduction ordering; conversely, if R is terminating, then the relation $\rightarrow_R^+$ itself is a reduction ordering. As suggested in [Manna-Ness, 1970], it is oftentimes convenient to separate reduction orderings into a homomorphism τ from the ground terms $\mathcal{G}(\mathcal{F})$ to an $\mathcal{F}$-algebra $\mathbf{W}$ and a "standard" well-founded ordering $\succ$ on $\mathbf{W}$. The homomorphism and ordering are constrained to satisfy the following monotonicity condition: $f_\tau(\ldots x \ldots) \succ f_\tau(\ldots y \ldots)$ whenever $x \succ y$, for all f in $\mathcal{F}$ and all x, y, etc. in W. Then, the ordering $\succ_\tau$, under which $s \succ t$ if $\tau(s) \succ \tau(t)$, for s and t in $\mathcal{G}$, is well-founded. To compare *free* (*open*) terms s and t in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, variables are added to W, and variables in terms are mapped by τ to distinct variables in $W(\mathcal{X})$. Then, $s \succ_\tau t$ only if $\tau(s) \succ \tau(t)$ for all assignments of values in W to the variables in $\tau(s)$ and $\tau(t)$. A system is terminating iff such $\mathbf{W}$, τ, and $\succ$ exist. For example, the system below, which computes the disjunctive normal form of formulae, can be shown terminating [Filman, 1978] with an exponential mapping into the natural numbers:

$$\begin{array}{rcl} not(not(x)) & \rightarrow & x \\ not(or(x,y)) & \rightarrow & and(not(x), not(y)) \\ not(and(x,y)) & \rightarrow & or(not(x), not(y)) \\ and(x, or(y,z)) & \rightarrow & or(and(x,y), and(x,z)) \\ and(or(y,z), x) & \rightarrow & or(and(y,x), and(z,x)) \end{array}$$

The mapping $\tau : \mathcal{T} \rightarrow \{2, 3, \ldots\}$ is defined by:

$$\begin{array}{rclrcl} or_\tau(a,b) & = & a + b + 1 & not_\tau(a) & = & 2^a \\ and_\tau(a,b) & = & a \cdot b & c_\tau & = & 2 \end{array}$$

for all a and b in $\{2, 3, \ldots\}$ and constant c in $\mathcal{F}_0$, with numbers compared under their natural ordering $>$. The second rule, for instance, always decreases the interpretation of a term, since $\tau(not(or(x,y))) = 2^{x+y+1}$ is greater than $\tau(and(not(x), not(y))) = 2^{x+y}$, for all x and y. A class of exponential interpretations were used for termination arguments in [Iturriaga, 1967].

The use, in particular, of *polynomial interpretations* was developed in [Lankford, 1975; Lankford, 1979]. Here a multivariate integer polynomial $f_\tau(x_1, \ldots, x_n)$ of degree n is associated with each n-ary symbol f in $\mathcal{F}_n$, for all n. The choice of coefficients must satisfy the monotonicity condition and ensure that terms are mapped into nonnegative integers only, as is the case, for example, when all coefficients are positive. Each rule must be shown to be reducing; that is, for each rule $l \rightarrow r$, the polynomial $\tau(l) - \tau(r)$ must be positive for all values of variables greater than the minimal value of a ground term (τ interprets variables in $\mathcal{X}$ as variables ranging over the naturals).

To prove termination of an associative-commutative class-rewriting system, the interpretation of an associative-commutative operator ought to be an associative-commutative polynomial. In general, such polynomials must be of either the quadratic form $f_\tau(x,y) = \alpha xy + \beta(x+y) + \beta(\beta-1)/\alpha$ ($\alpha \neq 0$) or the linear form $f_\tau(x,y) = x + y + \gamma$ (where α, β, and γ are natural numbers) [BenCherifa-Lescanne, 1987]. For example, one can use the following polynomial interpretation to prove termination of the BA/AC system of Section 2:

$$\begin{array}{rcl} xor_\tau(a,b) & = & a + b + 1 \\ and_\tau(a,b) & = & a \cdot b \\ c_\tau & = & 2. \end{array}$$

Primitive-recursive interpretations cannot suffice for termination proofs in general, since they would place a bound on the length of computations [Stickel, 1976]. In particular, integer polynomials place a double-exponential bound on the length of a derivation [Lautemann, 1988]. The following system, based on the "Battle of Hydra and Hercules" in [Kirby-Paris, 1982], is terminating, but not provably so in Peano Arithmetic:

$$\begin{array}{rcl}
h(z, e(x)) & \rightarrow & h(c(z), d(z, x)) \\
d(z, g(0,0)) & \rightarrow & e(0) \\
d(z, g(x, y)) & \rightarrow & g(e(x), d(z, y)) \\
d(c(z), g(g(x, y), 0)) & \rightarrow & g(d(c(z), g(x, y)), d(z, g(x, y))) \\
g(e(x), e(y)) & \rightarrow & e(g(x, y))
\end{array}$$

Think of $g(x, y)$ as the ordinal $\omega^x + y$, of $d(c^n(0), x)$ as any of the kth predecessors of x ($k \leq n$), and of $e(x)$ as x (e is just a place marker). Transfinite (ϵ_0-) induction is required for a proof of termination.

Nor do total reduction orderings suffice in termination arguments, as can be seen from the terminating system $\{f(a) \rightarrow f(b),\ g(b) \rightarrow g(a)\}$, for which a and b must be incomparable. Nevertheless, most of the orderings used in practice do extend to total reduction orderings. For a *total* rewrite ordering to be well-founded, it is necessary that it contain the proper subterm relation $\triangleright$, since if $t|_p \succ t$ for some term t and position p, then there is an infinite descending sequence $t \succ t[t]_p \succ t[t[t]_p]_p \succ \cdots$. As we will see shortly, for *finite* vocabularies, this "subterm" condition is also sufficient for well-foundedness of a rewrite ordering.

For termination of an ordered rewrite relation $\rightarrow_\succ$, the ordering $\succ$ according to which rewriting is performed must be a (well-founded) reduction ordering. Note that the variables occurring on the two sides of an equation need not coincide for termination, since the terms substituted for the variables must be such that the rewritten term is smaller vis-a-vis the reduction ordering. If the ordering $\succ$ is total on ground terms $\mathcal{G}$, then each ground instance of an equation in E can be oriented one way or another. As we will see, such ground orderings do exist. Of course, no rewrite ordering can be total on $\mathcal{T}$, since no two distinct variables x and y can be ordered (were we to have $x \succ y$, then we would have to have $y \succ x$, as well, the latter being an instance of the former).

5.2 Simplification Orderings

Termination arguments are often facilitated by the observation that all rewrite orderings containing the subterm relation $\trianglerighteq$ are well-founded (when $\mathcal{F}$ is finite). To see why the subterm property suffices, we need first to define a stronger notion than well-foundedness:

Definition 13. A quasi-ordering $\succeq$ on a set T is a *well-quasi-ordered* if every infinite sequence $t_1, t_2, \ldots$ of elements of T contains a pair of elements t_j and t_k, $j < k$, such that $t_j \preceq t_k$.

By the Pigeon-Hole Principle, an equivalence relation is a well-quasi-ordering iff there are only a finite number of equivalence classes.

The partial ordering $\succ$ associated with a quasi-ordering $\succeq$ is well-founded iff from some point on all elements in any infinite "quasi-descending" chain $t_1 \succeq t_2 \succeq t_3 \succeq \cdots$ are equivalent. Thus, any well-quasi-ordered set is well-founded (by the associated partial order), while a set is well-quasi-ordered if it is well-founded and has only a finite number of pairwise incomparable elements (the "finite anti-chain property"). It is also important to note that any extension of a well-quasi-ordering is a well-quasi-ordering and any restriction is well-founded (though it need not be well-quasi-ordered). This is what makes well-quasi-orderings convenient.

Our interest focuses on well-quasi-orderings of terms. Any well-quasi-ordering $\succeq$ on a vocabulary $\mathcal{F}$ induces a well-quasi-ordering $\succeq_{emb}$ on the terms $\mathcal{T}$ by means of the following set of schematic rules and equations:

$$\begin{array}{llll}
f(s_1, \ldots, s_n) & \rightarrow & s_i & 1 \leq i \leq n \\
f(s_1, \ldots, s_n) & \rightarrow & g(s_1, \ldots, s_n) & \text{if } f \succ g \\
f(s_1, \ldots, s_n) & \rightarrow & g(s_{i_1}, \ldots, s_{i_k}) & \text{if } f \succeq g,\ 1 \leq i_1 < \cdots < i_k \leq n,\ k < n \\
\hline
f(s_1, \ldots, s_n) & \leftrightarrow & g(s_1, \ldots, s_n) & \text{if } f \approx g
\end{array}$$

These schemata apply to all f and g of appropriate arity in $\mathcal{F}$. The first deletes context; the second decreases a function symbol; the third deletes subterms; the last replaces symbols with equivalents. (The third rule, with its condition changed to $k \leq n$, would subsume the second and fourth. As given, the termination of the one-way rules is more manifest.) We write $s \succeq_{emb} t$ if t is derivable from s using the above rules. The equivalence part of $\succeq_{emb}$ is just renaming symbols with equivalents under $\approx$. Viewing terms as (ordered) trees: $s \succeq_{emb} t$ if there is a mapping from the nodes in t into the nodes in s such that the function symbol labeling a node in t is less than or equivalent to (under $\succeq$) the label of the corresponding node in s, and such that distinct edges in t map to disjoint paths of s.

The following deep result is at the heart of our argument:

Tree Theorem ([Kruskal, 1960]). *If $\succeq$ is a well-quasi-ordering of a vocabulary $\mathcal{F}$, then the embedding relation $\succeq_{emb}$ is a well-quasi-ordering of the terms $\mathcal{T}(\mathcal{F})$.*

It has a beautiful proof, due to [Nash-Williams, 1963]:

Proof. Note that, by the infinite version of Ramsey's Theorem, any infinite sequence of elements of a well-quasi-ordered set must contain a subsequence that constitutes an infinite quasi-ascending chain.

Suppose, now, that the theorem were false. Then, there would exist (by the Axiom of Choice) a "minimal counter-example" sequence $t_1, t_2, \ldots, t_i, \ldots$, of which each element t_i is chosen so that it is smallest (in number of symbols) among all sequences of terms beginning with $t_1, t_2, \ldots, t_{i-1}$ and having no embedding $t_j \preceq_{emb} t_k$ for $j < k$. By the minimality hypothesis, the set of proper subterms of the elements of the minimal counter-example must be well-quasi-ordered (or else $t_1, t_2, \ldots, t_{l-1}, s_1, s_2, \ldots$ would be a smaller counter-example, where $s_1, s_2, \ldots$ is a counter-example of subterms of $t_l, t_{l+1}, \ldots$, such that s_1 is a subterm of t_l).

Since $\mathcal{F}$ is well-quasi-ordered by $\succeq$, there must exist an infinite subsequence $t_{i_1}, t_{i_2}, \ldots$ of the minimal counter-example such that their roots are a quasi-ascending chain. If any of these terms t_{i_j} are elements of $\mathcal{F}$, the original sequence could not have been a counter-example, because then $t_{i_j} \preceq_{emb} t_{i_{j+1}}$. Consider, then, the immediate subterms $w_{i_1}, w_{i_2}, \ldots$ of that subsequence. For example, if t_{i_j} is $f(g(a), b, g(b))$, then w_{i_j} is the word $g(a)\ b\ g(b)$. As noted above, the set of all these words must be well-quasi-ordered.

Using an auxiliary minimal counter-example argument, it can be shown that any infinite sequence of words over a well-quasi-ordered set contains a pair of words such that the first is a (not necessarily contiguous) subword of the second. (This result [Higman, 1952] is known as "Higman's Lemma.") In our case, this means that the infinite sequence of words composed of the immediate subterms of $t_{i_1}, t_{i_2}, \ldots$ must contain a pair w_{i_j} and w_{i_k} $(k > j)$ such that w_{i_j} is a subword of w_{i_k}. That, however, would imply that $t_{i_j} \preceq_{emb} t_{i_k}$, a contradiction. $\square$

The *(pure) homeomorphic embedding* relation $\to_{\trianglerighteq}^{*}$ is the special case of embedding, induced by simple equality of symbols for $\succeq$. In other words, it is derivability using only the first rule $f(s_1, \ldots, s_n) \to s_i$ of the previous system. It follows from the above theorem that any extension of homeomorphic embedding is a well-quasi-ordering of terms over a finite vocabulary. Since any rewrite ordering containing the subterm relation $\trianglerighteq$ also contains homeomorphic embedding, the subterm condition suffices for well-foundedness of term orderings over finite vocabularies, as claimed. Such orderings are the main tool for proving termination of rewriting:

Definition 14. A transitive and reflexive rewrite relation $\succeq$ is a *simplification ordering* if it contains the subterm ordering $\trianglerighteq$.

Simplification orderings are quasi-orderings (called "quasi-simplification orderings" in [Dershowitz, 1982a]) and are what Higman [Higman, 1952] called "divisibility orders." For finite R, only a finite number of function symbols can appear in any derivation $t_1 \to_R t_2 \to_R \cdots$. Thus, a finite R over $\mathcal{T}$ is terminating if there exists any simplification ordering $\succeq$ of $\mathcal{T}$ such that R is contained in its strict part $\succ$ [Dershowitz, 1982a]. The existence of such a simplification ordering means that $t_j \not\succeq t_k$ for all $k > j$, which precludes any t_j from being homeomorphically embedded in a subsequent t_k, as would necessarily be the case for any infinite derivation.

Virtually all the reduction orderings used in rewriting-system termination proofs are simplification orderings. For instance, integer polynomial interpretations with nonnegative coefficients are. One can even associate polynomials over the reals with function symbols and interpret terms as before [Dershowitz, 1979]. For a given choice τ of real polynomials to define a simplification ordering, $f_\tau(\ldots a \ldots) \geq a$ must always hold and $a \geq b$ must always imply $f_\tau(\ldots a \ldots) \geq f_\tau(\ldots b \ldots)$. For termination, $\tau(l)$ must be greater than $\tau(r)$ for each rule $l \rightarrow r$. All these inequalities need hold only when their variables are assigned values at least as large as the minimal interpretation of a constant, and are decidable [Tarski, 1951].

In difficult termination proofs, it is frequently useful to build more complicated orderings on top of simpler ones. For example, if $\succ_1$ and $\succ_2$ are partial orderings of S_1 and S_2, respectively, then we say that the pair $\langle s_1, s_2 \rangle$ is *lexicographically greater* than a pair $\langle s_1', s_2' \rangle$ (for s_1, s_1' in S_1 and s_2, s_2' in S_2), if $s_1 \succ_1 s_1'$, or else $s_1 = s_1'$ and $s_2 \succ_2 s_2'$. If $\succ_1$ and $\succ_2$ are well-founded, then the lexicographic ordering of the cross-product $S_1 \times S_2$ is also well-founded. In the same way, well-founded lexicographic orderings are defined on n-tuples of elements of well-founded sets.

Lexicographic orderings work for tuples of fixed length n. For collections of arbitrary size, another tool is needed. A *(finite) multiset* (or *bag*) is a finite unordered collection in which the number of occurrences of each element is significant. Formally, a multiset is a function from an element set S to the natural numbers, giving the multiplicity of each element. In general, if $\succ$ is a partial ordering on S, then the ordering $\succ_{mul}$ on multisets of elements of S is defined as the transitive closure of the replacement of an element with any finite number (including zero) of elements that are smaller under $\succ$. If $\succ$ is well-founded, the induced ordering $\succ_{mul}$ also is, as a consequence of König's Lemma for infinite trees [Dershowitz-Manna, 1979].

As an example of the application of lexicographic and multiset orderings to termination proofs, consider the rule:

$$x \cdot (y + z) \quad \rightarrow \quad (x \cdot y) + (x \cdot z)$$

We define a reduction ordering on terms as follows: Working our way from each innermost dot to the enclosing outermost dot, we construct a tuple of numbers, listing the size (total number of symbols) of the subterm headed by each dot encountered along the way. Each term is measured by the multiset of all its tuples (one for each innermost dot), with multisets compared in the ordering induced by the lexicographic ordering on tuples. The term $a \cdot ((b \cdot c) \cdot (d + (e \cdot f)))$, for example, is represented by $\{\langle 3, 9, 11 \rangle, \langle 3, 9, 11 \rangle\}$, while the term $a \cdot (((b \cdot c) \cdot d) + (b \cdot c) \cdot (e \cdot f))$ (after rewriting) is represented by $\{\langle 3, 5, 15 \rangle, \langle 3, 7, 15 \rangle, \langle 3, 7, 15 \rangle\}$. The latter multiset is smaller, since each of its elements is lexicographically smaller than $\langle 3, 9, 11 \rangle$, which appears in the former multiset (but not in the latter).

Multiset orderings will play an important role in Section 8.1.

5.3 Path Orderings

The above termination proof of the single distributivity rule is a complicated way of capturing the intuition that "$\cdot$" is, in some sense, the most significant function symbol. This suggests the possibility of constructing simplification orderings directly from well-founded orderings of vocabularies, or *precedences*. The idea [Plaisted, 1978; Dershowitz, 1982a] is that a term s should be bigger than any term that is built from terms smaller than s which are connected together by a structure of function symbols smaller, in the precedence, than the root of s. One such ordering is the "multiset path ordering" introduced in [Dershowitz, 1982a]:

Definition 15. For any given precedence $\succeq$, the *multiset path ordering* $\succeq_{mpo}$ is defined as derivability using the following schematic system *mpo*:

$$\begin{array}{rcll}
f(s_1, \ldots, s_n) & \rightarrow & s_i & 1 \leq i \leq n \\
f(s_1, \ldots, s_n) & \rightarrow & g(t_1, \ldots, t_m) & \text{if } f \succ g,\ f(s_1, \ldots, s_n) \rightarrow^+_{mpo} t_1, \ldots, t_m \\
f(s_1, \ldots, s_i, \ldots, s_n) & \rightarrow & g(s_1, \ldots, s_{i-1}, t_1, \ldots, t_k, s_{i+1}, \ldots, s_n) & \text{if } f \succeq g, \\
 & & & \quad s_i \rightarrow^+_{mpo} t_1, \ldots, s_i \rightarrow^+_{mpo} t_k,\ k \geq 0 \\
\hline
f(s_1, \ldots, s_n) & \leftrightarrow & g(s_{\pi_1}, \ldots, s_{\pi_n}) & \text{if } f \approx g,\ \pi \text{ a permutation}
\end{array}$$

The second rule replaces a term with one having a smaller root symbol. The third replaces a subterm with any number of smaller ones; in particular, it allows deletion of subterms ($k = 0$). Actually, the third rule must also permit any number of subterms to be replaced by smaller terms at the same time (to avoid violating the arity of symbols in the vocabulary).

The multiset path ordering contains the homeomorphic embedding relation and is, therefore, a simplification ordering. (That $\rightarrow^+_{mpo}$ is irreflexive is true, but not self-evident.) Moreover, if $\succ$ is a well-founded ordering of (possibly infinite) $\mathcal{F}$, then $\succ_{mpo}$ is a well-founded ordering of $\mathcal{T}$. To see this [Huet, 1980; Dershowitz, 1982a], note that (by Zorn's Lemma) a given precedence $\succ$ may be extended to an ordering $>$ such that the quasi-ordering $> \cup \approx$, call it $\gtrsim$, is a total well-quasi-ordering of $\mathcal{F}$. By Kruskal's Tree Theorem, the induced embedding relation $\gtrsim_{emb}$ well-quasi-orders $\mathcal{T}$, as does the total multiset path ordering $\gtrsim_{mpo}$ induced by the total precedence $\gtrsim$. Thus, $>_{mpo}$ is well-founded. Since the mapping from precedence $\succeq$ to term ordering $\succeq_{mpo}$ is *incremental*, in the sense that extending the precedence extends the corresponding ordering on terms, the smaller ordering $\succ_{mpo}$ must also be well-founded.

The multiset path ordering establishes termination of our stack and disjunctive normal form examples (as well as the bean-increasing Coffee Can Game). For the four-rule stack system, take the precedence *alternate* $>$ *push*. The first three rules are contained in $>_{mpo}$ by the first rule of *mpo*. For example, $pop(push(x, y)) \rightarrow_{mpo} push(x, y) \rightarrow_{mpo} y$. For the remaining stack rule, we have $alternate(push(x, y), z) >_{mpo} push(x, alternate(z, y))$, since *alternate* $>$ *push* and $alternate(push(x, y), z) \rightarrow^+_{mpo} x,\ alternate(z, y)$, the latter since $push(x, y) \rightarrow_{mpo} y$ and $alternate(y, z) \leftrightarrow_{mpo} alternate(z, y)$. Termination of the disjunctive normal form system may be shown using the precedence *not* $>$ *and* $>$ *or*.

One can think of the multiset path ordering as a functional mapping an ordering on function symbols (the precedence) to an ordering on terms, by recursively comparing the immediate subterms in the multiset extension of the term ordering. A related class of orderings [Kamin-Levy, 1980] compares subterms lexicographically, instead.

Definition 16. For any given precedence $\succeq$, the *lexicographic path ordering* $\succeq_{lpo}$ is defined as derivability by the following schematic system *lpo*:

$$\begin{array}{rcll}
f(s_1, \ldots, s_n) & \rightarrow & s_i & 1 \le i \le n \\
f(s_1, \ldots, s_n) & \rightarrow & g(t_1, \ldots, t_m) & \text{if } f \succ g,\ f(s_1, \ldots, s_n) \rightarrow^+_{lpo} t_1, \ldots, t_m \\
f(s_1, \ldots, s_i, \ldots, s_n) & \rightarrow & f(s_1, \ldots, s_{i-1}, t_i, \ldots, t_n) & \text{if } s_i \rightarrow^+_{lpo} t_i, \\
 & & & \qquad f(s_1, \ldots, s_n) \rightarrow^+_{lpo} t_{i+1}, \ldots, t_n \\
\hline
f(s_1, \ldots, s_n) & \leftrightarrow & g(s_1, \ldots, s_n) & \text{if } f \approx g
\end{array}$$

As in the multiset path ordering, the precedence $\succ$ induces an ordering on terms, but, here, subterms of the same function symbol are compared left-to-right, lexicographically. (They could just as well be compared right-to-left, or in any fixed order.)

The following traditional example—for Ackermann's function—illustrates its use with a precedence *ack* $>$ *succ*:

$$\begin{array}{rcl}
ack(0, y) & \rightarrow & succ(y) \\
ack(succ(x), 0) & \rightarrow & ack(x, succ(0)) \\
ack(succ(x), succ(y)) & \rightarrow & ack(x, ack(succ(x), y))
\end{array}$$

For example, the third rule is contained in $>_{lpo}$ since x occurs in $succ(x)$ and $ack(succ(x), succ(y))$ is lexicographically greater than $ack(succ(x), y)$.

If the strict part of a precedence is of order type α, then the multiset path ordering on the set of terms is of order type $\phi^\alpha(0)$ in the notation of [Feferman, 1968]. Combining multiset and lexicographic path orderings into one [Kamin-Levy, 1980], gives a more powerful ordering, which we call the *recursive path ordering* and which is related to Ackermann's ordinal notation [Dershowitz-Okada, 1988a]. (The original "recursive path ordering" [Dershowitz, 1982a] was of the multiset variety.) Determining if a precedence exists that makes two terms comparable is NP-complete [Krishnamoorthy-Narendran, 1984].

These precedence-based orderings are "syntactic", looking at function symbols one at a time. Similar semantically-oriented orderings have been devised; they replace the test $f \succ g$ by $s \succ t$, where $\succ$ is now a well-founded quasi-ordering of terms, not function symbols. For example, the *Knuth-Bendix ordering* [Knuth-Bendix, 1970] assigns a weight to a term which is the sum of the weights of its constituent function symbols. Terms of equal weight have their subterms compared lexicographically. Methods for choosing weights are described in [Lankford, 1979; Martin, 1987].

None of these orderings, however, can prove termination of the rule $f(f(x)) \rightarrow f(g(f(x)))$, since the right-hand side is embedded in the left. To overcome this problem, Puel [1989] compares "unavoidable patterns" instead of function symbols in the definition of $\succ_{mpo}$. By *unavoidable*, we mean that any sufficiently large term in $\mathcal{T}$ must be greater, under the encompassment ordering $\trianglerighteq$, than one of the unavoidable patterns. For example, any term constructed from a constant a and three or more f's and g's must contain an occurrence of one of the three patterns: $f(f(x))$, $f(g(x))$, or $g(g(x))$. The well-foundedness of this "pattern path ordering" is based on a powerful extension of Kruskal's Tree Theorem [Puel, 1989] (analogous to a similar theorem on strings in [Ehrenfeucht-etal, 1983]).

5.4 Combined Systems

We saw above that polynomial orderings are applicable to associative-commutative systems, but are severely restrictive. The multiset path ordering, though compatible with commutativity axioms, is not well-founded when associativity is added as a bi-directional rule to *mpo*. For example, let the precedence be $c \succ b$. Then, $(c+b)+b \rightarrow_{mpo} c+(b+b) \leftrightarrow_S (c+b)+b$, where S is associativity of $+$. To overcome this problem, the multiset path ordering has been adapted to handle associative-commutative operators by flattening and also transforming terms (distributing symbols that are bigger in the precedence over smaller ones) before comparing [Bachmair-Plaisted, 1985], i.e. s is greater than t iff the T-normal form $T(s)$ of s is greater under $\succ_{mpo}$ than the T-normal form $T(t)$ of t, for some convergent "transform" system T and precedence $\succ$. The general use of rewrite systems as transforms and the formulation of abstract conditions of the resultant reduction ordering are explored in [Bachmair-Dershowitz, 1986; Bellegarde-Lescanne, 1987].

Termination of the union of term- or class-rewriting systems can be reduced to the termination of each: Let R and S be two binary relations contained in well-founded orderings $\succ_R$ and $\succ_S$, respectively. If $\succ_S$ *commutes* over $\succ_R$, i.e. if $\succ_R \circ \succ_S$ is contained in $\succ_S \circ \succ_R$, then the union $R \cup S$ is terminating. Actually, it suffices if $\succ_S$ *quasi-commutes* over $\succ_R$, by which we mean that $\succ_R \circ \succ_S$ is contained in $\succ_S \circ \succ =_R$. For example, if R is any terminating rewrite system, then the union of $\rightarrow_R$ with the proper subterm relation $\triangleright$ is well-founded, since taking subterms is well-founded and if a subterm rewrites then so does the superterm. More generally, the union of any terminating rewrite relation with the (proper) encompassment ordering $\trianglerighteq$ is also well-founded, since encompassment is just $\geq$ (subsumption) and/or $\triangleright$, both of which are well-founded, and any rewrite relation commutes over both. Similarly, if R is a binary relation contained in a well-founded ordering $\succ_R$, S is a symmetric binary relation contained in a congruence $\approx_S$, and $\succ_R$ commutes over $\approx_S$, then the composite relation $R \circ S$ is terminating. To prove termination of a combined term-rewriting system $R \cup S$, it is necessary and sufficient that R and S be contained in reduction orderings that commute as above; to prove termination of a class-rewriting system R/S, it is necessary and sufficient that R be contained in a reduction ordering that commutes over a symmetric and transitive rewrite relation that contains S. These ideas generalize results in [Bachmair-Dershowitz, 1986; Jouannaud-Munoz, 1984]. Note that commutation of $\rightarrow_R$ and $\rightarrow_S$ is *not* ensured by R and S having disjoint vocabularies, the system at the beginning of this section being a counter-example [Toyama, 1987b].

5.5 Further Reading

Martin Gardner [1983] talks about multiset orderings and the Hydra battle. For a survey of the history and applications of well-quasi-orderings, see [Kruskal, 1972]. For a comprehensive survey of termination, see [Dershowitz, 1987]. The multiset and lexicographic path orderings, and their variants (see [Rusinowitch, 1987]), have been implemented in many rewriting-rule based theorem provers (e.g. [Lescanne, 1984]). Some results on the complexity of derivations appear in [Choppy-etal, 1987].

6 SATISFIABILITY

We turn our attention now to the determination of satisfiability. If an equation $s = t$ is satisfiable in the (free-) term algebra $\mathcal{T}$, that is, if $s\sigma$ and $t\sigma$ are identical for some substitution σ, then s and t are said to be *unifiable*. The unification problem, *per se*, is to determine if two terms are unifiable. More particularly, we are interested in determining the set of all unifying substitutions σ. Though unifiable terms may have an infinite number of unifiers, there is—as we will see—a unique substitution (unique up to literal similarity) that is minimal with respect to the subsumption ordering on substitutions. More generally, we are interested in solving equations in the presence of equational axioms specifying properties of the operators. For a given equational theory E, we say that s and t are *E-unifiable* if $s = t$ is satisfiable in the free quotient algebra $\mathcal{T}/E$, in which case it is satisfiable in all models of E. In general, there may be no minimal solution to a given E-unification problem.

6.1 Syntactic Unification

Let $\doteq$ be literal similarity of terms, under which two terms s and t are equivalent if each is an instance of the other. We show that the quotient $\mathcal{T}/\doteq$, ordered by the subsumption relation $\dot{\geq}$, is a lower semi-lattice, by showing that every pair of terms, s and t, has a greatest lower-bound, $glb(s,t)$, called their *least (general) generalization* [Plotkin, 1972]. (Note that $\mathcal{T}/\doteq$ is not an algebra, because literal similarity is not a congruence.) Let LG be the following set of "transformation" rules, operating on pairs $(P; w)$, where w is a term containing the partial solution, and P contains the pairs yet to be solved:

$$\begin{array}{lr}
\text{Decompose:} & (\{f(s_1,\ldots,s_m) \sqcap_x f(t_1,\ldots,t_m)\} \cup P \;;\; w) \quad \Rightarrow \\
& (\{s_1 \sqcap_{x_1} t_1, \ldots, s_n \sqcap_{x_n} t_n\} \cup P \;;\; w\sigma) \\
& \text{where } \sigma \text{ is } \{x \mapsto f(x_1,\ldots,x_n)\} \\
& \text{and } x_1,\ldots,x_n \text{ are distinct new variable symbols} \\
\text{Coalesce:} & (\{s \sqcap_x t,\; s \sqcap_y t\} \cup P \;;\; w) \quad \Rightarrow \quad (\{s \sqcap_y t\} \cup P \;;\; w\sigma) \\
& \text{where } \sigma \text{ is } \{x \mapsto y\}
\end{array}$$

Each pair is written as $s \sqcap_x t$, where x is a variable of w: Applying these rules to $(\{s \sqcap_x t\} \;;\; x)$, where x is not a variable of s or t, until none is applicable, results in $(\{u_i \sqcap_\xi v_i\} \;;\; glb(s,t))$. To prove that repeated applications of LG always terminate, note that each application of a rule decreases the number of function symbols (not including $\sqcap$) in P. Since the system $\Rightarrow_{LG}$ is actually Church-Rosser (on $\mathcal{T}/\doteq$), least generalizations are unique up to literal similarity. For example, the least generalization of $f(g(a), g(b), a)$ and $f(g(b), g(a), b)$ is $f(g(x), g(y), x)$. It follows from properties of well-founded lattices [Birkhoff, 1948] that every pair of terms s and t that are bounded from above (i.e., there exists a term that is an instance of both) have a least (i.e. most general) upper-bound, denoted $lub(s,t)$.

An equation $s = t$ has a *solution* σ if $s\sigma = t\sigma$. Here, s and t may share variables and we demand that applying a single substitution σ (mapping all occurrences of the same variable to the same term) result in identical terms. The least solution with respect to subsumption, $mgu(s,t)$, is called their *most general unifier*. For example, the most general common instance of $alternate(y', \Lambda)$ and $alternate(push(x,y), z)$ is $alternate(push(x,y), \Lambda)$, or anything literally similar; the mgu of $alternate(y, \Lambda)$ and $alternate(\Lambda, z)$ is $\{y \mapsto \Lambda, z \mapsto \Lambda\}$; there is no solution to $alternate(z, \Lambda) = alternate(push(x,y), z)$. Most general unifiers and least upper bounds of terms are closely related: by revising s and t so that the two terms have disjoint variables, we get $lub(s,t) = s\mu$, where $\mu = mgu(s', t')$, for literally similar terms s' and t' of s and t, respectively; in the other direction, $s\mu$ and $t\mu$ are both equal to $lub(eq(x,x), eq(s,t))$, where $\mu = mgu(s,t)$, eq is any binary symbol, and x is any variable. As a consequence, the most general unifier is unique up to literal similarity, but need not always exist. This fundamental uniqueness result for first-order terms does not hold true for higher-order languages with function variables [Huet, 1976].

Robinson [1965] was the first to give an algorithm for finding most general unifiers. Following [Herbrand, 1930; Martelli-Montanari, 1982], we view unification as a step-by-step process of transforming multisets of equations, until a "solved form" is obtained from which the most general unifier can be extracted. A *solved form* is any set of equations $\{x_1 = s_1, \ldots, x_n = s_n\}$ such that the x_i are distinct and no x_i is a variable in any s_j. Then, the most general unifier is the substitution $\{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}$. Let $|t|$ denote the *size* of the term t, that is the total number of its function symbols. (The size of a variable is zero.) Equations to be solved will be written in the form $s =^? t$. Define a well-founded ordering $\succ$ on equations as follows: $u =^? v \succ s =^? t$ if (i) $max(|u|, |v|) > max(|s|, |t|)$, or else $max(|u|, |v|) = max(|s|, |t|)$ and $max(|u|, |v|) - min(|u|, |v|)$ is greater than $max(|s|, |t|) - min(|s|, |t|)$. We also use a constant F to denote the absence of a solution, and make it smaller than any equation. Let MM be the following set of *transformation rules* operating on pairs $(P; S)$ of sets of equations, with P containing the equations yet to be solved and S, the partial solution:

$$
\begin{array}{lrcl}
\text{Delete:} & (\{s =^? s\} \cup P; S) & \Rightarrow & (P; S) \\
\text{Decompose:} & (\{f(s_1, \ldots, s_m) =^? f(t_1, \ldots, t_m)\} \cup P; S) & \Rightarrow & (\{s_1 =^? t_1, \ldots, s_n =^? t_n\} \cup P; S) \\
\text{Fail:} & (\{f(s_1, \ldots, s_m) =^? g(t_1, \ldots, t_n)\} \cup P; S) & \Rightarrow & (\emptyset; \{F\}) \\
 & & & \text{if } f \neq g \\
\text{Merge:} & (\{x =^? s, x =^? t\} \cup P; S) & \Rightarrow & (\{x =^? s, s =^? t\} \cup P; S) \\
 & & & \text{if } x \in \mathcal{X} \text{ and } x =^? t \succ s =^? t \\
\text{Coalesce:} & (\{x =^? y\} \cup P; S) & \Rightarrow & (P\sigma; S\sigma \cup \{x = y\}) \\
 & & & \text{if } x, y \in \mathcal{X} \text{ and } x \neq y, \text{ where } \sigma = \{x \mapsto y\} \\
\text{Check:} & (\{x =^? s\} \cup P; S) & \Rightarrow & (\emptyset; \{F\}) \\
 & & & \text{if } x \in \mathcal{X},\ x \text{ occurs in } s, \text{ and } x \neq s \\
\text{Eliminate:} & (\{x =^? s\} \cup P; S) & \Rightarrow & (P\sigma; S\sigma \cup \{x = s\}) \\
 & & & \text{if } x \in \mathcal{X},\ s \notin \mathcal{X}, \text{ and } x \text{ does not occur in } s, \text{ where } \sigma = \{x \mapsto s\}
\end{array}
$$

Definition 17. A *(syntactic) unification procedure* is any program that takes a finite set P_0 of equations, and uses the above rules MM to generate a sequence of inferences from $(P_0; \emptyset)$.

Starting with $(\{s =^? t\}; \emptyset)$ and using the unification rules repeatedly until none is applicable, results in $(\emptyset; \{F\})$ iff $s =^? t$ has no solution, or else it results in a solved form $(\emptyset; \{x_1 = s_1, \ldots, x_n = s_n\})$. The application of any of these rules does not change the set of solutions of $P \cup S$. Hence, the former signifies failure, and in the latter case, $\sigma = \{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}$ is a most general unifier of s and t. That σ is most general follows from the fact that the rules preserve all solutions.

For example, the most general unifier of $f(x, x, a)$ and $f(g(y), g(a), y)$ is $\{x \mapsto g(a), y \mapsto a\}$, since

$$
\begin{array}{rcll}
(\{f(x, x, a) =^? f(g(y), g(a), y)\}; \emptyset) & \Rightarrow_{MM} & (\{x =^? g(y), x =^? g(a), a =^? y\}; \emptyset) & \Rightarrow_{MM} \\
(\{x =^? g(y), g(y) =^? g(a), a =^? y\}; \emptyset) & \Rightarrow_{MM} & (\{x =^? g(a), g(a) =^? g(a)\}; \{y = a\}) & \Rightarrow_{MM} \\
(\{x =^? g(a)\}; \{y = a\}) & \Rightarrow_{MM} & (\emptyset; \{x = g(a), y = a\}) &
\end{array}
$$

On the other hand, $f(x, x, x)$ and $f(g(y), g(a), y)$ are not unifiable, since

$$
\begin{array}{rcll}
(\{f(x, x, x) =^? f(g(y), g(a), y)\}; \emptyset) & \Rightarrow_{MM} & (\{x =^? g(y), x =^? g(a), x =^? y\}; \emptyset) & \Rightarrow_{MM} \\
(\{y =^? g(y), y =^? g(a)\}; \{x = y\}) & \Rightarrow_{MM} & (\emptyset; \{F\}) &
\end{array}
$$

on account of an "occur check."

To prove that repeated applications of MM always terminate, we can use a lexicographic combination of an ordering on numbers and the multiset extension of the $\succ$ ordering on equations. With each application of a rule, $(P; S) \Rightarrow_{MM} (P'; S')$, either the solved set S is enlarged, or the problem set P is reduced under $\succ_{mul}$. Since the solved set cannot increase without bound (it can have at most one equation per variable), nor can the unsolved set decrease without limit (since $\succ_{mul}$ is well-founded), there can be no infinite MM-derivations. Uncontrolled use of **eliminate** leads to exponential time complexity. With appropriate data structures and control strategies, an efficient algorithm is obtained, which is quasi-linear in the worst case (e.g. [Baxter, 1976]); more careful implementations provide for truly linear, but less practical, algorithms (e.g [Paterson-Wegman, 1978]). The **coalesce** rule allows us to avoid the use of "multi-equations"; cf. [Martelli-Montanari, 1982]. Eliminating the **check** rule produces solutions over the domain of (infinite) "rational" trees [Huet, 1976], and has ramifications for the semantics of some Prolog implementations [Colmerauer, 1984].

6.2 Semantic Unification

When it comes to E-unification, the situation is much more complex. A substitution σ is a *solution in E* to an equation $s = t$ if $s\sigma =_E t\sigma$, in which case we say that σ is an *E-unifier* of s and t; we say that t *E-matches* s if there exists a substitution σ such that $s\sigma =_E t$. E-unifiability is undecidable whenever the word problem is, and in many other cases as well. For example, the solvability of Diophantine equations, that is, polynomial equations over the integers, is undecidable [Matijasevic, 1970], as is unifiability under associativity and distributivity alone [Szabo, 1982]). Satisfiability may be undecidable even even when congruence classes are finite (as for associativity, commutativity, and distributivity [Siekmann, 1984]). Second-order unifiability (equivalence of function definitions) is also undecidable, in general [Goldfarb, 1981]. On the brighter side, many other theories have decidable unification problems, including Presburger arithmetic [Presburger, 1927; Shostak, 1979], real closed fields [Tarski, 1951; Collins, 1975] and monoids [Makanin, 1978].

When more than one solution may exist for a theory E, we define a solution σ to be *more general* than a solution ρ, if $\sigma \lessdot_E \rho$ in the E-subsumption ordering $\lessdot_E$, i.e. if there exists a substitution τ such that $x\sigma\tau =_E x\rho$, for all variables x in $\mathcal{X}$, but not vice-versa. An E-unifier is *most general* if no more general unifier exists. Note that E-subsumption is not well-founded for all E. There are decidable theories with infinite sets of most general unifiers (an example is the set of solutions $\{a^i | i \geq 1\}$ to $x \cdot a = a \cdot x$, where "$\cdot$" is associative [Plotkin, 1972]), and there are some for which there are solutions, but no most general one [Fages-Huet, 1983] (an example is associativity plus idempotence [Baader, 1986]). We say that a set S of E-unifiers is *complete* if for every E-unifier there is one in S that is more general with respect to E-subsumption. For example, a complete unification algorithm exists for associativity and commutativity (AC) [Stickel, 1981; Herold-Siekmann, 1987]; alternative algorithms with better performance are [Kirchner, 1989; Boudet, 1989]. Other theories for which algorithms are available that compute finite, complete sets of most general E-unifiers include commutativity, AC with identity and/or idempotency (see [Fages, 1987]), as well as Boolean rings (see [Boudet-etal, 1988]). For many of these theories, unification is believed intractable from the time-complexity point of view [Kapur-Narendran, 1986].

Of course, E-unifiability is semi-decidable for recursively-enumerable E. Paramodulation (without the functional reflexivity axioms) [Robinson-Wos, 1969] is one improvement over the obvious "British-museum" method of interleaving the production of substitutions with the search for equational proofs.

Paramodulation may be improved upon by a more goal-oriented process. The following set of rules, inspired by [Gallier-Snyder, 1987; Hsiang-Jouannaud, 1988], does the trick:

$$
\begin{array}{lrcl}
\text{Decompose:} & (\{f(s_1,\ldots,s_m) \leftrightarrow^? f(t_1,\ldots,t_m)\} \cup P; S) & \Rightarrow & (\{s_1 \leftrightarrow^? t_1,\ldots,s_n \leftrightarrow^? t_n\} \cup P; S) \\
\text{Eliminate:} & (\{x \leftrightarrow^? s\} \cup P; S) & \Rightarrow & (P\sigma; S\sigma \cup \{x = s\}) \\
& \multicolumn{3}{r}{\text{if } x \in \mathcal{X}, \text{ and } x \text{ does not occur in } s, \text{ where } \sigma = \{x \mapsto s\}} \\
\text{Mutate:} & (\{f(s_1,\ldots,s_n) \leftrightarrow^? t\} \cup P; S) & \Rightarrow & (\{s_1 \leftrightarrow^? u_1,\ldots,s_n \leftrightarrow^? u_n, r \leftrightarrow^? t\}; S) \\
& \multicolumn{3}{r}{\text{if } f(u_1,\ldots,u_n) = r \text{ is literally similar to an equation in } E} \\
& \multicolumn{3}{r}{\text{but has no variables in common with } S,\ P,\ t, \text{ or the } s_i} \\
\text{Splice:} & (\{s \leftrightarrow^? t\} \cup P; S) & \Rightarrow & (\{s \leftrightarrow^? x, r \leftrightarrow^? t\} \cup P; S) \\
& \multicolumn{3}{r}{\text{if } x = r \text{ is literally similar to an equation in } E} \\
& \multicolumn{3}{r}{\text{but has no variables in common with } S,\ P,\ t, \text{ or the } s_i} \\
\text{Imitate:} & (\{f(s_1,\ldots,s_n) \leftrightarrow^? y\} \cup P; S) & \Rightarrow & \\
& \multicolumn{3}{r}{(\{s_1 \leftrightarrow^? y_1,\ldots,s_n \leftrightarrow^? y_n, y \leftrightarrow^? f(y_1,\ldots,y_n)\} \cup P; S)} \\
& \multicolumn{3}{r}{\text{if } y \in \mathcal{X} \text{ and the } y_i \text{ are new variables}}
\end{array}
$$

We call this set of rules EU. These rules *non-deterministically* compute solved forms, each of which represents an E-unifier.

Definition 18. An *E-unification procedure* is any (non-deterministic) program that takes a finite set P_0 of equations, and uses the above rules EU to generate sequences of inferences from $(P_0; \emptyset)$.

To generate a complete set of unifiers may not require computing all possible sequences. In particular, the use of **imitate** can be severely restricted [Hsiang-Jouannaud, 1988]. The set of rules EU can be improved for particular classes of equations. Two special cases have been investigated: (i) when E is "strict" in the sense that for any valid equation $s = t$, there exists a proof with at most one proof step taking place at the top position [Kirchner, 1986]; and (ii) when there exists a ground-convergent set of rules complete for E. In the first case, axioms of the form $x = t$, for variable x, are disallowed; hence **splice** is superfluous. Moreover, **decompose** must always apply after **mutate**; hence, the two can be compiled into a single rule. For example, commutativity uses the following "mutate and decompose" rule:

$$(\{f(s_1, s_2) =^? f(t_1, t_2)\} \cup P; S) \quad \Rightarrow \quad (\{s_1 =^? t_2, s_2 =^? t_1\} \cup P; S)$$

More complex cases, involving bounded applications of axioms in E prior to decomposition, can be treated similarly.

Methods of combining unification algorithms for well-behaved theories that do not share symbols have been given in [Yelick, 1987; Kirchner, 1989; Boudet-etal, 1988]; The general case was solved in [Schmidt-Schauss, 1988]. Note that a unification algorithm that generates a complete set of most general unifiers (for terms without free constants) does not automatically work for matching (one cannot just treat the variables of t as constants, since that changes the algebra and may introduce unsound solutions) [Burckert-etal, 1987].

6.3 Narrowing

Even when a convergent system R exists for a theory E, E-unification problem remains only semi-decidable. For example, the system

$$\begin{array}{rcl|rcl|rcl}
x + 0 & \rightarrow & x & x + succ(y) & \rightarrow & succ(x + y) & x + pred(y) & \rightarrow & pred(x + y) \\
x - 0 & \rightarrow & x & x - succ(y) & \rightarrow & pred(x - y) & x - pred(y) & \rightarrow & succ(x - y) \\
x * 0 & \rightarrow & 0 & x * succ(y) & \rightarrow & (x * y) + y & x * pred(y) & \rightarrow & (x * y) - y \\
succ(pred(x)) & \rightarrow & x & pred(succ(x)) & \rightarrow & x & & &
\end{array}$$

for addition and multiplication of integers is canonical, but were R-unification (or R-matching) decidable, then the existence of integer solutions to Diophantine equations, such as

$$x \cdot x \;+\; y \cdot y \;-\; succ(succ(succ(0))) \quad = \quad 0,$$

would also be decidable. The latter is Hilbert's Tenth Problem, shown to be undecidable in [Matijasevic, 1970]. (Cf. [Bockmayr, 1987; Heilbrunner-Holldobler, 1987].)

When a convergent R is available, a one-way sort of paramodulation suffices, due to the existence of a rewrite proof for an arbitrary valid equation [Dershowitz-etal, 1987b; Martelli-etal, 1989]. The following set of rules, RU, restricts uses of equations to left-hand sides of rules:

$$\begin{array}{lrcl}
\text{Decompose:} & (\{f(s_1, \ldots, s_m) \rightarrow^? f(t_1, \ldots, t_m)\} \cup P; S) & \Rightarrow & (\{s_1 \rightarrow^? t_1, \ldots, s_n \rightarrow^? t_n\} \cup P; S) \\
\text{Eliminate:} & (\{x \rightarrow^? s\} \cup P; S) & \Rightarrow & (P\sigma; S\sigma \cup \{x = s\}) \\
& \multicolumn{3}{r}{\text{if } x \in \mathcal{X}, \text{ and } x \text{ does not occur in } s, \text{ where } \sigma = \{x \mapsto s\}} \\
\text{Mutate:} & (\{f(s_1, \ldots, s_n) \rightarrow^? t\} \cup P; S) & \Rightarrow & (\{s_1 \rightarrow^? u_1, \ldots, s_n \rightarrow^? u_n, r \rightarrow^? t\}; S) \\
& \multicolumn{3}{r}{\text{if } f(u_1, \ldots, u_n) \rightarrow r \text{ is literally similar to a rule in } R} \\
& \multicolumn{3}{r}{\text{but has no variables in common with } S, P, t, \text{ or the } s_i} \\
\text{Imitate:} & (\{f(s_1, \ldots, s_n) \rightarrow^? y\} \cup P; S) & \Rightarrow & \\
& \multicolumn{3}{r}{(\{s_1 \rightarrow^? y_1, \ldots, s_n \rightarrow^? y_n, y \rightarrow^? f(y_1, \ldots, y_n)\} \cup P; S)} \\
& \multicolumn{3}{r}{\text{if } y \in \mathcal{X} \text{ and the } y_i \text{ are new variables}}
\end{array}$$

This set of rules subsumes "narrowing", as used for this purpose in [Fay, 1979]:

Definition 19. A term s *narrows* (in one step) to a term t, *via* substitution μ, symbolized $s \rightsquigarrow_R t$, if t is $s\mu[r\mu]_p$, for some non-variable position p in s, rule $l \rightarrow r$ in R (the variables of which have been renamed so that they are distinct from those in s), and most general unifier μ of $s|_p$ and l.

The verb "narrow" perhaps carries the wrong connotation: it is the set of R-congruence classes of instances of the term that is being narrowed. It is easy to see that RU mimics narrowing by using **decompose**, **imitate**, and **mutate**. For example, if R is $\{f(x,x) \rightarrow c(x), a \rightarrow b\}$, then the problem $\{f(a,y) \rightarrow^? z,\ f(y,b) \rightarrow^? z\}$ mutates to $\{a \rightarrow^? x,\ y \rightarrow^? x,\ c(x) \rightarrow^? z,\ f(y,b) \rightarrow^? z\}$. Then a is imitated by x, and x and y are eliminated, by substituting a for them: $\{c(a) \rightarrow^? z,\ f(a,b) \rightarrow^? z\}$. This corresponds to narrowing of $f(a,y)$ to $c(a)$ by instantiating $y \mapsto a$, and eventually yields the solution $\{y \mapsto a, z \mapsto c(b)\}$.

Narrowing has the following property:

Narrowing Lemma. *If R is a convergent rewrite system and $s\sigma \rightarrow_R^* t$, then there exist terms u and v such that $s \rightsquigarrow_R^* u$, $t \rightarrow_R^* v$, and $v \succeq u$.*

If σ is irreducible (that is, if $x\sigma$ is irreducible for all variables x), then $t = v$, and the lemma holds even for non-convergent R [Hullot, 1980]. Without convergence, reducible solutions are lost. For example, if R is $\{f(a,b) \rightarrow c, a \rightarrow b\}$ or $\{f(x,g(x)) \rightarrow c, a \rightarrow g(a)\}$, then $f(y,y)$ cannot be narrowed, and $f(y,y) \rightarrow^? c$ fails to lead to a solved form, despite the fact that there is a solution $\{y \mapsto a\}$.

Variations on narrowing include: *normal narrowing* [Fay, 1979] (in which terms are normalized via $\rightarrow_R^!$ before narrowing), *basic narrowing* [Hullot, 1980] (in which the substitution part of prior narrowings is not subsequently narrowed), and their combination [Rety-etal, 198?], all of which are semi-complete for convergent R. Class-rewriting yields similar results [Jouannaud-etal, 1983].

6.4 Further Reading

For a survey regarding syntactic unification, see [Lassez-etal, 1988]; for unification in general, see [Huet, 1976]. For a survey of theory and applications of syntactic and semantic unification, see [Knight, 1989]. Questions of decidability of unification in equational theories are summarized in [Siekmann, 1984]; a summary of complexity results for some of the decidable cases is [Kapur-Narendran, 1987]. A popular exposition on the undecidability of the existence of solutions to Diophantine equations is [Davis-Hersh, 1973]. For a comprehensive treatment of narrowing and E-unification, see [Kirchner, 1985]. For the satisfiability problem of arbitrary first-order formulae with equality as the only predicate, see [Maher, 1988],Comon-Lescanne-1989.

7 CRITICAL PAIRS

In this section, we continue our study of the Church-Rosser property for rewrite systems. In particular, we will see that confluence is decidable for finite, terminating systems. Confluence, in general, is undecidable [Huet, 1980], even if all rules are monadic [Book-etal, 1981]. For finite ground systems—even if they are nonterminating—decision procedures exist (see [Dauchet-etal, 1987; Oyamaguchi, 1987]). Ground confluence, on the other hand, is undecidable, even if the system is terminating [Kapur-etal, 1987]. Even for convergent systems R, the questions whether congruence classes defined by $\leftrightarrow_R^*$ are finite in number, or are all finite in size, are undecidable, unless R is ground [Raoult, 1981].

7.1 Term Rewriting

Let $l \rightarrow r$ and $s \rightarrow t$ be two rules. We say that the left-hand side l *overlaps* the left-hand side s if there is a *nonvariable* subterm $s|_p$ of s such that l and $s|_p$ have a common upper bound with respect to subsumption. To determine overlap, the variables in the two (not necessarily distinct) rules are renamed, if necessary, so that they are disjoint. Then, l overlaps s if there exists a unifying substitution σ such that $l\sigma = s\sigma|_p$. When there is an overlap, the overlapped term $s\sigma$ can be rewritten to either $t\sigma$ or $s\sigma[r\sigma]_p$. The two-step proof $t\sigma \leftarrow_R s\sigma[l\sigma]_p \rightarrow_R s\sigma[r\sigma]_p$ is called a *critical peak*.

Figure 4: Proof of Critical Pair Lemma.

Definition 20. If $l \to r$ and $s \to t$ are two rewrite rules with distinct variables, p is the position of a nonvariable subterm of s, and μ is a most general unifier of $s|_p$ and l, then the equation $t\mu = s\mu[r\mu]_p$ is a *critical pair* formed from those rules.

Thus, a critical pair is the equation arising from a most general nonvariable overlap between two left-hand sides. For example, $push(x, alternate(\Lambda, y)) = push(x, y)$ is the critical pair obtained from $alternate(push(x, y), z) \to push(x, alternate(z, y))$ and $alternate(y, \Lambda) \to y$.

Let $\mathbf{cp}(R)$ denote the set of all critical pairs between (not necessarily distinct, but perhaps renamed) rules in R and let $\leftrightarrow_{\mathbf{cp}(R)}$ denote its symmetric rewrite closure.

Critical Pair Lemma ([Knuth-Bendix, 1970]). *For any rewrite system R and peak $s \leftarrow_R u \to_R t$, there either exists a rewrite proof $s \to_R^* v \leftarrow_R^* t$ or a critical-pair proof $s \leftrightarrow_{\mathbf{cp}(R)} t$.*

The proof [Knuth-Bendix, 1970], depicted in Figure 4, considers all relative positions of the two redexes.

As stressed in [Huet, 1980], no assumption of termination is necessary for this lemma. It follows from this and the Diamond Lemma, that a terminating system R is confluent iff $\mathbf{cp}(R)$, regarded as a relation, is a subset of the joinability relation $\to_R^* \circ \leftarrow_R^*$. This holds, for instance, for the stack interleaving example of Section 3.2. Since finite systems have a finite number of critical pairs, their confluence is decidable, provided they are terminating. This criterion for confluence is called the *superposition test* [Knuth-Bendix, 1970].

Without termination, a system may have no critical pairs (hence be locally confluent), and still be non-confluent. A system *sans* critical pairs (except trivial ones of the form $t = t$) is called *non-overlapping*, or *non-ambiguous*. The following example of a non-overlapping, but non-confluent system [Huet, 1980], is based on the "ladder" in Figure 2(b):

$$\begin{array}{rcl} f(x, x) & \to & a \\ f(x, g(x)) & \to & b \\ c & \to & g(c) \end{array}$$

The term c has no normal form, but $f(c, c)$ has two, a and b. This example can be modified so that the system is normalizing [Sivakumar, 1986]:

$$
\begin{array}{rcl}
f(x,x) & \rightarrow & g(x) \\
f(x,g(x)) & \rightarrow & b \\
h(c,y) & \rightarrow & f(h(y,c),h(y,y))
\end{array}
$$

The Critical Pair Lemma can be weakened so that not all pairs need be considered. Various such *critical pair criteria* have been investigated, all revolving around the case of a critical peak that is rewritable in additional ways, making it possible to replace the peak with an alternate proof that is in some sense smaller. The *connectedness* criterion is based on the well-founded method of establishing local confluence (mentioned in Section 4.1) and ignores any critical pair $s = t$ derived from an overlap $s \leftarrow_R u \rightarrow_R t$ such that there exists another proof $s \leftrightarrow_R^* t$, each term of which is derivable from u by $\rightarrow_R^+$ [Winkler-Buchberger, 1983]; the *compositeness* criterion ignores critical pairs for which the overlapped term can be rewritten at a position strictly below the point of overlap [Kapur-etal, 1988].

7.2 Regular Systems

By enforcing strong restrictions on the form of left-hand sides, confluence can be ensured even for non-terminating systems.

Definition 21. A rewrite system that is both left-linear (no multiple occurrences of a variable on the left) and non-overlapping (no non-trivial critical pairs) is called *regular*.

Examples of regular systems are the stack and Combinatory Logic systems (of Sections 2.3 and 2.4, respectively). Mutually recursive function definitions, with one equation (employing *if* $\cdot$ *then* $\cdot$ *else* $\cdot$) per defined function, are regular. Regular systems are always confluent, by the following result:

Parallel Moves Lemma ([Huet, 1980]). *If R is regular, then $\rightarrow_R^{\|}$ is strongly confluent.*

The symbol $\rightarrow_R^{\|}$ denotes one "parallel" application of rules in R at disjoint redexes. (For the name and inspiration of this lemma, cf. [Curry-Feys, 1958]). The confluence of regular systems establishes the consistency of the operational semantics of recursive programming languages; see [Raoult-Vuillemin, 1980]. The above lemma may be weakened to allow critical pairs that join in one parallel step [Huet, 1980]; the ground case was considered in [Rosen, 1973].

For regular systems, normal forms can be computed by a "parallel-outermost" redex evaluation scheme [O'Donnell, 1977a], but not by a "leftmost-outermost" scheme [Huet-Levy, 1990]; with additional "sequentiality" requirements, one can efficiently compute normal forms, without lookahead [Hoffmann-O'Donnell, 1979; Huet-Levy, 1990]. How to avoid all unnecessary rewrites and obtain an optimal strategy is, however, an undecidable problem, in general [Huet-Levy, 1990].

For modularity of programming with rewrite systems, one would have wished that the union of two convergent systems over disjoint vocabularies be convergent. Unfortunately, though the union of two confluent systems sharing no function symbols is confluent [Toyama, 1987a], the termination property (as we saw in Section 5.4) is not preserved under disjoint union. This is true even for confluent systems [Toyama, 1987b]. If, however, the two convergent systems are also left-linear, then their union is convergent [Toyama-etal, 1989].

7.3 Class Rewriting

Critical pairs also provide a necessary and sufficient condition for a left-linear terminating system R to be Church-Rosser modulo a congruence S. That is, if $\mathbf{cp}(R \cup S)$ is a subset of $\rightarrow_R^! \circ \leftrightarrow_S^* \circ \leftarrow_R^!$, then $\leftrightarrow_{R \cup S}^*$ is also contained therein [Huet, 1980]. Then, R-normal forms may be used to decide validity, provided S-equivalence is decidable. For example, if S is commutativity and R includes all commutativity variants of its rules, then R-normal forms are unique up to permutations of operands.

To handle rewriting modulo a congruence in the presence of non-left-linear rules, [Peterson-Stickel, 1981] suggested using the extended rewrite relation $S\backslash R$ to compute normal forms. The set of critical peaks of the form $t\mu \leftarrow_{S\backslash R} s\mu \rightarrow_{S\backslash R} s\mu[r\mu]_p$ is in general infinite, so the Critical Pair Lemma is of little practical help. Instead, we consider peaks $t\mu \leftarrow_R s\mu \rightarrow_{S\backslash R} s\mu[r\mu]_p$ and *cliffs* $t\mu \leftrightarrow_S s\mu \rightarrow_{S\backslash R} s\mu[r\mu]_p$ separately.

Definition 22. Let S be an set of equations. If $s \rightarrow t$ and $l \rightarrow r$ are two rewrite rules with distinct variables, p is the position of a nonvariable subterm of s, and μ is a most general substitution (most general, with respect to subsumption modulo S) such that $s\mu|_p =_S l\mu$, then $t\mu = s\mu[r\mu]_p$ is an *S-critical pair* of the two rules. If $s \leftrightarrow t$ is an equation in S, $l \rightarrow r$ is a rewrite rule (renamed as necessary), p is the position of a nonvariable proper subterm of s, and μ is a most general substitution such that $s\mu|_p =_S l\mu$, then $t\mu \rightarrow s\mu[r\mu]_p$ is an *S-extended rule* of $l \rightarrow r$.

In the set $\mathbf{cp}_S(R)$, we include all critical pairs obtained by overlapping S-variants of rules in R on (renamed) rules in R. We also need the set $\mathbf{ex}_S(R)$ of extended rules obtained by overlapping variants of rules in R on (renamed) equations in S.

Extended Critical Pair Lemma ([Jouannaud, 1983]). *For any rewrite system R, equational system S, and peak $s \leftarrow_R \circ \rightarrow_{S\backslash R} t$, there exists a rewrite proof $s \rightarrow^*_{S\backslash R} \circ \leftrightarrow^*_S \circ \leftarrow^*_{S\backslash R} t$, or a critical-pair proof $s \leftrightarrow^*_S \circ \leftrightarrow_{\mathbf{cp}_S(R)} \circ \leftrightarrow^*_S t$ which may involve S-steps within the critical pair's variable part only. Similarly, for any cliff $s \leftrightarrow_S \circ \rightarrow_{S\backslash R} t$, there exists a rewrite proof $s \rightarrow^*_{S\backslash R} \circ \leftrightarrow^*_S \circ \leftarrow^*_{S\backslash R} t$, or an extended-rule proof $s \leftrightarrow^*_S \circ \rightarrow_{\mathbf{ex}_S(R)} \circ \leftrightarrow^*_S t$ which may involve S-steps within the extended rule's variable part only.*

The point is that in the absence of a rewrite proof, there must be a proof that is an application of an "S-instance" of a critical pair. The possible need for S-steps in the variable part is illustrated by the equation $a \leftrightarrow b$ and rules $f(x) \rightarrow g(x)$ and $f(x) \rightarrow h(x)$. A critical peak $g(a) \leftarrow_R f(a) \leftrightarrow_S f(b) \rightarrow_R h(b)$ lends itself to the critical pair proof $g(a) \leftrightarrow_S g(b) \leftrightarrow_{\mathbf{cp}(R)} h(b)$.

Using this lemma, it can be shown that if R/S is terminating and the subterm relation modulo S is well-founded, then $S\backslash R$ is Church-Rosser modulo S iff $\mathbf{cp}_S(R)$ and $\mathbf{ex}_S(R)$ are contained in $\rightarrow^!_{S\backslash R} \circ \leftrightarrow^*_S \circ \leftarrow^!_{S\backslash R}$ [Jouannaud-Kirchner, 1986]. If these conditions are satisfied, and an S-matching procedure is available, then validity in $R \cup S$ can be decided. Note that subterm modulo S is well-founded when S-congruence classes are finite.

It is possible to combine the above results by partitioning rules into left-linear and not necessarily left-linear subsets. The critical pair condition can then be tailored to the different kinds of rules, with term rewriting used for the left-linear subset and extended rewriting for the rest [Jouannaud-Kirchner, 1986]. Additional improvements are provided by critical pair criteria for extended rewriting, as described in [Bachmair-Dershowitz, 1987a].

7.4 Ordered Rewriting

Ordered rewriting systems enjoy a similar critical pair condition for confluence, but only for certain classes of orderings and only for ground terms. An ordering $>$ is called a *complete simplification ordering* if it is a simplification ordering that is total on $\mathcal{G}$, i.e. for any two distinct ground terms u and v, either $u > v$ or $v > u$. For example, if a precedence is total, it is easy to show that the induced lexicographic path ordering $>_{lpo}$ is a complete simplification ordering (a property not shared by the multiset path ordering).

Definition 23. Let $l = r$ and $s = t$ be two equations in E (with disjoint variables) such that l overlaps s at non-variable position p with most general unifier μ. The equation $t\mu = s\mu[r\mu]_p$ is a *critical pair* if the participating steps $t\mu \leftrightarrow_E s\mu \leftrightarrow_E s\mu[r\mu]_p$ can form a peak; in other words, if $t\mu\gamma \leftarrow_E s\mu\gamma \rightarrow_E s\mu\gamma[r\mu\gamma]_p$, for some substitution γ.

Let $\mathbf{cp}_>(E)$ denote the set of all such critical pairs between equations in E.

Ordered Critical Pair Lemma ([Lankford, 1975]). *For any set of equations E, complete simplification ordering $>$, and peak $s \leftarrow_E u \rightarrow_E t$ between ground terms s, t, u, there either exists an ordered-rewrite proof $s \rightarrow^*_E \circ \leftarrow^*_E t$ or a critical-pair proof $s \leftrightarrow_{\mathbf{cp}_>(E)} t$.*

The difference between the proof of this and the original Critical Pair Lemma is only that the bottom step in Figure 4(b) may go one way or the other, depending on whether $s > t$ or vice-versa. (If $s = t$, the rewrite proof is trivial.) It is to ensure that s and t are comparable, that we require $>$ to be total on ground terms.

It may seem that local confluence is not guaranteed by the critical pair condition, when equations in E have variables on one side that do not appear on the other. For example, with $E = \{x + g(y) \ = \ x + f(z)\}$, and $>$ the lexicographic path ordering induced by the precedence $g \ > \ f \ > \ b \ > \ a$, there is a peak $a + f(b) \leftarrow_E a + g(a) \rightarrow_E a + f(a)$, whereas there is no rewrite proof $a + f(b) \rightarrow_E a + f(a)$. But, in fact, there is a critical pair hidden here: overlapping the left-hand side of the equation on a literally similar instance $x + g(y') \ = \ x + f(z')$ gives the pair $x + f(z) \ = \ x + f(z')$. Thus, $a + f(b) \longrightarrow_{\mathbf{cp}_>(E)} a + f(a)$.

With local confluence established, we have a ground convergent rewrite relation $\rightarrow_E$. Such an ordered-rewriting system—when finite—may be used to decide validity: if $s \ = \ t$ is valid for ground convergent E, then by *Skolemizing* the variables in s and t, that is by treating those variables as constants, and extending the ordering $>$ to include them, both s and t will have the same normal form. For example, if E is the commutativity axiom $x \cdot y = y \cdot x$ and $z \ > \ y \ > \ x$ in a lexicographic path ordering, then $(y \cdot x) \cdot z$ and $z \cdot (y \cdot x)$ have the same normal form, $(x \cdot y) \cdot z$.

As a more interesting example, consider the following system for entropic groupoids [Hsiang-Rusinowitch, 1987]:

$$\begin{array}{rcl} (x \cdot y) \cdot x & \rightarrow & x \\ x \cdot (y \cdot z) & \rightarrow & x \cdot z \\ ((x \cdot y_1) \cdot y_2) \cdot z) & \rightarrow & x \cdot z \\ \hline (x \cdot y_1) \cdot z & \leftrightarrow & (x \cdot y_2) \cdot z \end{array}$$

and suppose we wish to decide validity of an equation $s = t$. First, the variables $x_1, \ldots, x_n$ appearing in s and t are replaced by Skolem constants $c_1, \ldots, c_n$. Then, a lexicographic path ordering is used with a precedence in which "$\cdot$" is larger than the constants, and the constants are linearly ordered: $c_n > \ldots > c_1$. The equation is used to rewrite any product of the form $(x \cdot y_1) \cdot z$ to the same term with the occurrence of y_1 replaced by the smallest term (viz. c_1) under $>_{lpo}$.

7.5 Reduced Systems

By reducing right-hand sides and deleting rules with rewritable left-hand sides, a convergent system can always be converted into a canonical, i.e. reduced and convergent, one (see, e.g., [Metivier, 1983]). One of the nice things about reduced systems is that, for any given equational theory, there can be only one (finite or infinite) canonical system contained in a particular reduction ordering [Butler-Lankford, 1980; Metivier, 1983]. This uniqueness result is up to literal similarity. Uniqueness does not, however, hold for arbitrary canonical class-rewriting systems [Dershowitz-etal, 1988], but does for associative-commutative systems [Lankford-Ballantyne, 1983].

7.6 Further Reading

A detailed study of the Church-Rosser property of non-overlapping systems is [Klop, 1980]. Computing with regular systems is the subject of [O'Donnell, 1977a].

8 COMPLETION

In the previous section, we saw that confluence of finite terminating systems can be decided using the superposition test. Suppose a given system fails that test because some critical pair has no rewrite proof. Building on ideas of Evans [1951], Knuth and Bendix [1970], suggested extending such a system with a new rule tailored to cover the offending critical pair. Of course, new rules mean new critical pairs, some of which may also not pass the test. But, often enough, repeating this process eventually leads to a convergent system, with all critical pairs having rewrite proofs. This procedure is called *completion*. Interestingly, the critical pairs generated along the way are frequently the kind of lemmata a mathematician would come up with [Knuth-Bendix, 1970].

Starting with a finite set of equations and a reduction ordering on terms, the completion procedure attempts to find a finite canonical system for the theory presented by the equations by generating critical pairs and orienting them as necessary. If reducing the two sides of a critical pair $s = t$ yields an equation $u = v$, where u and v are not identical, then adding a new rule $u \rightarrow v$ or $v \rightarrow u$ supplies a rewrite proof for $s = t$. To decide between the two orientations, the given reduction ordering is employed: if $u \succ v$ then $u \rightarrow v$ is added, while if $v \succ u$ then $v \rightarrow u$ is chosen. The new rule, $u \rightarrow v$ or $v \rightarrow u$, is then used to form new critical pairs. Running the procedure can have one of three outcomes: success in finding a canonical system, failure in finding anything, or looping and generating an infinite number of rules (forming an infinite canonical system).

8.1 Abstract Completion

Completion has recently been put in a more abstract framework [Bachmair-etal, 1986], an approach we adopt here. As in traditional proof theory (cf. [Takeuti, 1987]), proofs are reduced, in some well-founded sense, by replacing locally maximal subproofs with smaller ones, until a normal-form proof is obtained. In completion, the axioms used are in a constant state of flux; these changes are expressed as inference rules, which add a dynamic character to establishing the existence of reducible subproofs. This view of completion, then, has two main components: an inference system, used in the completion process to generate new rewrite rules, and a rewrite relation that shows how any proof can be normalized, as long as the appropriate rules have been generated.

An *inference rule* (for our purposes) is a binary relation between pairs $(E; R)$, where E is a set of equations and R is a set of rewrite rules. (Rules or equations that differ only in the names of their variable are, for all intents and purposes, treated as identical.) Let $\succ$ be a reduction ordering, and $\succeq$ the well-founded ordering on rules defined as follows: $s \rightarrow t \succeq l \rightarrow r$ if (i) $s \trianglerighteq l$ under the encompassment ordering, or else (ii) $s \doteq l$ (s and l are literally similar) and $t \succ r$. We define the following set KB of six inference rules:

$$
\begin{array}{llcll}
\text{Delete:} & (E \cup \{s = s\}; R) & \vdash & (E; R) & \\
\text{Compose:} & (E; R \cup \{s \rightarrow t\}) & \vdash & (E; R \cup \{s \rightarrow u\}) & \text{if } t \rightarrow_R u \\
\text{Simplify:} & (E \cup \{s = t\}; R) & \vdash & (E \cup \{s = u\}; R) & \text{if } t \rightarrow_R u \\
\text{Orient:} & (E \cup \{s = t\}; R) & \vdash & (E; R \cup \{s \rightarrow t\}) & \text{if } s \succ t \\
\text{Collapse:} & (E; R \cup \{s \rightarrow t\}) & \vdash & (E \cup \{u = t\}; R) & \\
 & & & \multicolumn{2}{l}{\text{if } s \rightarrow_R u \text{ by } l \rightarrow r \text{ with } s \rightarrow t \succeq l \rightarrow r} \\
\text{Deduce:} & (E; R) & \vdash & (E \cup \{s = t\}; R) & \text{if } s = t \in \mathbf{cp}(R)
\end{array}
$$

We write $(E; R) \vdash_{KB} (E'; R')$ if the latter may be obtained from the former by one application of a rule in KB. **Delete** removes a trivial equation $s = s$. **Compose** rewrites the right-hand side t of a rule $s \rightarrow t$, if possible. **Simplify** rewrites either side of an equation $s = t$. **Orient** turns an equation $s = t$ that is orientable ($s \succ t$ or $t \succ s$) into a rewriting rule. **Collapse** rewrites the left-hand side of a rule $s \rightarrow t$ and turns the result into an equation $u = t$, but only when the rule $l \rightarrow r$ being applied to s is smaller than the rule being removed under the rule ordering. **Deduce** adds equational consequences to E, but only those that follow from critical overlaps $s \leftarrow_R u \rightarrow_R t$.

Definition 24. A *(standard) completion procedure* is any program that takes a finite set E_0 of equations and a reduction ordering $\succ$, and uses the above rules KB to generate a sequence of inferences from $(E_0; \emptyset)$.

In practice, the completion rules are usually applied in the given order, saving space by preserving only reduced rules and equations. The *results* of a finite completion sequence $(E_0; \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots \vdash_{KB} (E_n; R_n)$ are E_n and R_n; more generally, the *limit* of a possibly infinite completion sequence $(E_0; \emptyset) \vdash_{KB}$

$(E_1; R_1) \vdash_{KB} \cdots$ is the pair $(E_\infty; R_\infty)$, where E_∞ is the set $\cup_{i\geq 0} \cap_{j\geq i} E_j$ of *persisting* equations and R_∞ is the set $\cup_{i\geq 0} \cap_{j\geq i} R_j$ of *persisting* rules. We say that a completion sequence is *successful*, if E_∞ is empty and R_∞ is canonical.

If success occurs after a finite number of steps, then the resultant system R_∞ is a decision procedure for E_0. But the completion may "loop", producing an infinitely large set of persisting rules. A simple example [Ardis, 1980] of looping is provided by the equation $f(g(f(x))) = g(f(x))$. Oriented the only way possible, the rule $f(g(f(x))) \rightarrow g(f(x))$ overlaps itself, generating the critical pair $g(f(g(f(x)))) = f(g(g(f(x))))$, which simplifies to $g(g(f(x))) = f(g(g(f(x))))$. Continuing in the same manner, an infinite set of rules $\{f(g^i(f(x))) \rightarrow g^i(f(x)) | i \geq 1\}$ is produced.

The rules in KB are evidently *sound*, in that the class of provable theorems is unchanged by an inference step. Furthermore, only rules contained in $\succ$ are added. We are thus assured that the limit R_∞ of any (finite or infinite) successful completion sequence is terminating and presents the same equational theory as did E_0. Figure 5 shows an example of a successful completion sequence. Starting with the three axioms

$$\begin{array}{rcl} x \cdot 1 & = & x \\ 1 \cdot x & = & x \\ x^- \cdot (x \cdot y) & = & y \end{array}$$

over a vocabulary containing a constant 1, postfix unary symbol "$^-$", and infix binary symbol "$\cdot$", it generates the eight-rule canonical system

$$\begin{array}{rcl|rcl} 1 \cdot x & \rightarrow & x & x \cdot 1 & \rightarrow & x \\ x^- \cdot x & \rightarrow & 1 & x \cdot x^- & \rightarrow & 1 \\ 1^- & \rightarrow & 1 & (x^-)^- & \rightarrow & x \\ x^- \cdot (x \cdot y) & \rightarrow & y & x \cdot (x^- \cdot y) & \rightarrow & y \end{array}$$

using size as the reduction ordering.

For a given reduction ordering $\succ$, a (not necessarily finite) convergent system R contained in $\succ$ exists for an equational theory E, iff each E-congruence class of terms has a unique minimal element vis-a-vis $\succ$ [Avenhaus, 1985]. Nonetheless, completion may fail to find R, even when given $\rightarrow_R^+$ as the reduction ordering [Dershowitz-etal, 1988]. For example, despite the existence of $\{f(a) \rightarrow a, c \rightarrow a, b \rightarrow a\}$, no successful sequence exists for $\{f(b) = a, f(c) = c, b = c\}$, as long as b and c are incomparable under the given ordering. In fact, on account of the partialness of the ordering, some sequences may fail while others may succeed [Avenhaus, 1985; Dershowitz-etal, 1988]. For example, let $\succ$ be a recursive path ordering with precedence $f \succ d \succ c \succ a$ and $d \succ b \succ a$ (but b and c are incomparable), and let $E_0 = \{f(c) = c, b = d, c = d, f(d) = a\}$. There is a successful sequence:

$$(E_0; \emptyset) \overset{+}{\underset{KB}{\vdash}} (\{b = d, f(d) = a\}; \{d \rightarrow c, f(c) \rightarrow c\}) \overset{+}{\underset{KB}{\vdash}} (\emptyset; \{f(a) \rightarrow a, b \rightarrow a, c \rightarrow a, d \rightarrow a\})$$

as well as a failing one:

$$(E_0; \emptyset) \overset{+}{\underset{KB}{\vdash}} (\{c = d, f(d) = a\}; \{d \rightarrow b, f(c) \rightarrow c\}) \overset{+}{\underset{KB}{\vdash}} (\{b = c\}; \{f(b) \rightarrow a, d \rightarrow b, f(c) \rightarrow c\})$$

The latter sequence cannot be extended further.

As pointed out already in [Knuth-Bendix, 1970], such failures can be circumvented by incorporating an inference rule that adds $s \rightarrow k(x_1, \ldots, x_n)$ and $t \rightarrow k(x_1, \ldots, x_n)$ to R_i if $s = t$ is an unorientable equation in E_i, where k is a new function symbol not in the original vocabulary and $x_1, \ldots, x_n$ are those variables appearing in s and t. Though this inference is not sound (it constitutes a conservative extension), it results in a decision procedure if ultimately successful. In the above failing example, replacing $b = c$ with $b \rightarrow k$ and $c \rightarrow k$ leads directly to $\{a \rightarrow k, b \rightarrow k, c \rightarrow k, d \rightarrow k, f(k) \rightarrow k\}$. Two terms s and t in $\mathcal{T}(\{a, b, c, d, f\})$ are equal in the original theory iff they have the same normal form in this system. Unfortunately, this process can, in

i	R_i	E_i	inference
0		$1 \cdot x = x$	
		$x \cdot 1 = x$	
		$x^- \cdot (x \cdot y) = y$	
1	$x \cdot 1 \rightarrow x$	$1 \cdot x = x$	orient
		$x^- \cdot (x \cdot y) = y$	
2	R_1	$x^- \cdot (x \cdot y) = y$	orient
	$1 \cdot x \rightarrow x$		
3	R_2		orient
4	$x^- \cdot (x \cdot y) \rightarrow y$	$x^- \cdot x = 1$	deduce (1,3)
5	R_3		orient
6	$x^- \cdot x \rightarrow 1$	$1^- = 1$	deduce (1,5)
7	R_5		orient
8	$1^- \rightarrow 1$	$(x^-)^- \cdot y = x \cdot y$	deduce (3,3)
9	R_7		orient
10	$(x^-)^- \cdot y \rightarrow x \cdot y$	$1^- \cdot y = 1 \cdot y$	deduce (7,9)
11	$1^- \cdot y \rightarrow 1 \cdot y$		orient
12	R_9		compose (11,2)
	$1^- \cdot y \rightarrow y$		
13	R_9	$1 \cdot y = y$	collapse (12,7)
14		$y = y$	simplify (2)
15			delete
16		$(x^-)^- = x$	deduce (1,9)
17	R_9		orient
	$(x^-)^- \rightarrow x$		
18	R_7	$x \cdot y = x \cdot y$	collapse (9,17)
19	$1^- \cdot y \rightarrow y$		delete
20	$(x^-)^- \rightarrow x$	$x \cdot (x^- \cdot y) = y$	deduce (3,17)
21	R_{18}		orient
22	$x \cdot (x^- \cdot y) \rightarrow y$	$x \cdot x^- = 1$	deduce (1,21)
23	R_{21}		orient
	$x \cdot x^- \rightarrow 1$		

Table 1: A successful completion sequence for a fragment of group theory.

general, degenerate into unsuccessful sequences that add infinitely many new symbols. As we will see below, completion has been extended in various ways, in particular to handle the associative-commutative axioms, which cannot be handled as rules.

Completion can also be used as a mechanical theorem prover. The idea is that, even when the procedure loops, any valid theorem should eventually have a rewrite proof using the rules already on hand. This is not actually the case, since a procedure might abort when no equation is orientable. But for implementations that are fair in their choice of inferences, one can show that—barring abortion—all provable equations eventually lend themselves to a direct proof by rewriting. This perspective on completion was taken by Huet [Huet, 1981], and in [Lankford, 1975] from the refutational point of view. The difficult part is the need to show that deleting simplifiable rules does not—in the long run—shrink the class of equations having rewrite proofs. Once we have established that completion acts as a semi-decision procedure for validity when it loops, we will be ready to apply it to theorem-proving in inductive theories and in first-order predicate calculus.

A proof in $E \cup R$ is a sequence of E-steps and R-steps. By applying the above inference rules, it may be possible to simplify a given proof, replacing some steps with alternate ones from $E' \cup R'$, whenever $(E; R) \vdash^*_{KB} (E'; R')$. By a *proof pattern* we intend a schema describing a class of subproofs; e.g. to characterize rewrite proofs, we use the pattern $s \rightarrow^*_R v \leftarrow^*_R t$, where s, t, and v denote arbitrary terms. If a proof contains no peaks $s \leftarrow_R u \rightarrow_R t$ nor applications $s \leftrightarrow_E t$ of equations, it must be a rewrite proof. Let E^* be the set of all equations deducible from E_0 and R^* be the orientable subset of E^* that intersects with the reduction ordering $\succ$. The following set C of proof-pattern rules captures the elimination of the undesirable patterns and the simplification of proofs that takes place during completion:

$$
\begin{array}{rcll}
s \leftrightarrow_{E^*} s & \Rightarrow & s & \\
s \rightarrow_{R^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftarrow_{R^*} t & \text{where } s \rightarrow_{R^*} t \text{ by } l \rightarrow r \text{ and } s \rightarrow_{R^*} v \text{ by } l' \rightarrow r' \text{ and } l \rightarrow r \succeq l' \rightarrow r' \\
s \leftrightarrow_{E^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftrightarrow_{E^*} t & \\
s \leftrightarrow_{E^*} t & \Rightarrow & s \rightarrow_{R^*} t & \\
s \rightarrow_{R^*} t & \Rightarrow & s \rightarrow_{R^*} v \leftrightarrow_{E^*} t & \text{where } s \rightarrow_{R^*} t \text{ by } l \rightarrow r \text{ and } s \rightarrow_{R^*} v \text{ by } l' \rightarrow r' \text{ and } l \rightarrow r \succeq l' \rightarrow r' \\
s \leftarrow_{R^*} u \rightarrow_{R^*} t & \Rightarrow & s \leftrightarrow_{E^*} t & \\
s \leftarrow_{R^*} u \rightarrow_{R^*} t & \Rightarrow & s \rightarrow^*_{R^*} v \leftarrow^*_{R^*} t &
\end{array}
$$

Symmetric rules, with the proof patterns on both sides of $\Rightarrow$ inverted, are also needed.

Note how these rules correspond exactly to the effect of the inference rules: Any proof step involving a deleted equation can be omitted; when a rule is replaced with a new composed rule, any rewrite step using it can be replaced by a two-step valley; when an equation is simplified, its use in a proof is replaced by a rewrite step and a smaller equational step; when an equation is oriented, the corresponding proof step becomes a rewrite step; when a rule is collapsed into an equation, a combination of a rewrite and equational steps may be used instead; when a critical pair is deduced, the corresponding critical peak in a proof may be replaced by a new equational step. The last proof-pattern rule corresponds to a non-critical peak, which can always be converted into a valley.

We assume that the ordering is such that for any equation $s = t$ in E^* there is a term v for which $s \rightarrow^*_{R^*} v \leftarrow^*_{R^*} t$. (With a reduction ordering that does not satisfy this condition there is no chance of a successful completion.) Then, at least one of the rules of C can be applied to any non-rewrite proof or to any proof employing a non-reduced rule. Thus, C-normal forms are R^*-rewrite proofs that use only reduced rules. Furthermore, we can apply the techniques of Section 5 and show that the proof-normalization relation $\Rightarrow_C$ is terminating: Consider the ordering $\succ_c$ which compares proofs by comparing multisets containing the pair $\langle \{s\}, l \rightarrow r \rangle$ for each application $s \rightarrow_{R^*} t$ of a rule $l \rightarrow r$ and $\langle \{s, t\}, l \rightarrow r \rangle$ for each application $s \leftrightarrow_{E^*} t$ of an equation $l = r$. Pairs are compared lexicographically, using the multiset ordering $\succ_{mul}$ induced by the given reduction ordering $\succ$ for the first component, and the ordering $\succeq$ on rules for the second. Multisets of pairs, measuring the complexity of proofs, are compared by $\succ_c$, the multiset ordering induced by this lexicographic ordering. Since $\succ$ and $\succeq$ are both well-founded, $\succ_c$ is a reduction ordering on *proofs*. Since it can be verified that C is contained in $\succ_c$, the former is terminating. Note how this *proof ordering* considers the justification of a proof and not just the terms in it. For further details, consult [Bachmair, 1989b].

8.2 Fairness

For any given completion sequence $(E_0; \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots$, let $\leftrightarrow_i^*$ stand for $\leftrightarrow_{E_i \cup R_i}^*$, that is, for a proof at the ith stage, using rewriting with R_i in either direction or equational steps with E_i. The proof normalization relation C mirrors the inference system KB in that for any completion sequence and for any proof $s \leftrightarrow_i^* t$ at stage i there exists a proof $s \leftrightarrow_j^* t$ at each subsequent stage j such that $s \leftrightarrow_i^* t \Rightarrow_C^* s \leftrightarrow_j^* t$. In this way, inference rules are used to generate rules needed for proofs to be C-reducible. A (possibly infinite) sequence $(E_0; \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots$ is deemed *fair* if for any proof $s \leftrightarrow_i^* t$ that is C-reducible, there exists a step j, such that $s \leftrightarrow_i^* t \Rightarrow_C^+ s \leftrightarrow_j^* t$. Imposing fairness, we have:

Proof Normalization Theorem ([Huet, 1981]). *If a completion sequence* $(E_0; \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots$ *is fair, then for any proof* $s \leftrightarrow_i^* t$ *there exists a proof* $s \rightarrow_{R_\infty}^* v \leftarrow_{R_\infty}^* t$ *using reduced rules only.*

Huet introduced the notion of fairness of completion and proved this theorem for a specific fair implementation; the following proof [Bachmair-etal, 1986] builds on the above development and holds for any implementation of the inference system KB:

Proof. The proof is by induction with respect to $\Rightarrow_C$. Suppose that $s \leftrightarrow_i^* t$ is not a rewrite proof $s \rightarrow_{R_\infty}^* \circ \leftarrow_{R_\infty}^* t$. Then it must contain a peak that is reducible by C. Similarly, if $s \leftrightarrow_i^* t$ involves a nonpersistent step, then it is C-reducible. By fairness, $s \leftrightarrow_i^* t \Rightarrow_C^+ s \leftrightarrow_j^* t$ for some step j and by induction there is a proof $s \rightarrow_{R_\infty}^* \circ \leftarrow_{R_\infty}^* t$ with only reduced rules. □

By the Critical Pair Lemma, non-critical peaks are C-reducible. Thus, it can be shown that a completion sequence is fair if all persistent critical pairs are accounted for ($\mathbf{cp}(R_\infty)$ is a subset of $\cup E_i$), no simplifiable rule persists (R_∞ is reduced), and no equation persists (E_∞ is empty). Practically speaking, fairness means that critical pairs are generated for all new rules, and need eventually to be simplified or oriented, unless the new rule itself is later simplified. A marking scheme is generally used to keep track of which rules still need to be overlapped with which; see, for instance, [Huet, 1981]. By generating the remaining critical pairs and then eliminating them, Figure 5 becomes fair.

We say that an n-step completion sequence $(E_0; \emptyset) \vdash_{KB} (E_1; R_1) \vdash_{KB} \cdots \vdash_{KB} (E_n; R_n)$ *succeeds* if E_n is empty, R_n is reduced, and each of the latter's critical pairs already appeared in some E_i. The sequence *fails* if no fair sequence has it as a prefix; in that case there is little point continuing. A completion procedure is considered *correct* if it only generates fair, successful sequences—when it does not abort. Assuming the procedure never discriminates against any critical pair or simplifiable rule or equation, the only good reason to abort (and produce a failing sequence) is if all equations are unorientable. The critical pair criteria mentioned in the Section 7.1 may be used to improve on the above requirements for fairness, by necessitating consideration of fewer critical pairs; see [Bachmair-Dershowitz, 1988].

It follows from the above theorem that the limit R_∞ of a fair sequence is canonical. Furthermore, since $\leftrightarrow_\infty^* = \leftrightarrow_0^*$, completion, correctly implemented, provides a semi-decision procedure for validity, or else aborts. That is, if $\mathcal{M}od(E_0) \models s = t$, and if completion does not give up in the middle, then, at some step k, it will be possible to check that $s \rightarrow_{R_k}^! \circ \leftarrow_{R_k}^! t$.

Note that two successful derivations, given the same starting set E_0 and reduction ordering $\succ$, must output the same canonical systems, be they finite or infinite, since there can be but one reduced, canonical system (up to literal similarity) contained in $\succ$ (see Section 7.5). Thus, if there exists a canonical system R for E_0 contained in $\succ$, then a correct procedure, given E_0 and $\succ$, cannot succeed with any system but R—though it may abort without finding it. Furthermore, if R is finite, then an infinite, looping completion sequence is likewise impossible, since R_∞ must be of the same size as R.

8.3 Extended Completion

Before we consider other theorem-proving applications of completion, we adapt it to handle extended rewriting. Let S be an equational system and $\succ$ a reduction ordering such that $\succ$ commutes over S. Rules are compared using the following ordering: $s \rightarrow t \succeq l \rightarrow r$ if $s \trianglerighteq s' =_S l$, for some s' (i.e. if s properly encompasses a term that is S-equivalent to l), or else $s \doteq l$ and $t \succ r$. We define the following set KB/S of inference rules:

$$
\begin{array}{llcll}
\text{Delete:} & (E \cup \{s = t\}; R) & \vdash & (E; R) & \text{if } s \leftrightarrow_S^* t \\
\text{Compose:} & (E; R \cup \{s \rightarrow t\}) & \vdash & (E; R \cup \{s \rightarrow v\}) & \text{if } t \rightarrow_{R/S} v \\
\text{Simplify:} & (E \cup \{s = t\}; R) & \vdash & (E \cup \{u = t\}; R) & \text{if } s \rightarrow_{R/S} u \\
\text{Orient:} & (E \cup \{s = t\}; R) & \vdash & (E; R \cup \{s \rightarrow t\}) & \text{if } s \succ t \\
\text{Collapse:} & (E; R \cup \{s \rightarrow t\}) & \vdash & (E \cup \{v = t\}; R) & \\
 & & & \multicolumn{2}{l}{\text{if } s \rightarrow_{R/S} v \text{ by } l \rightarrow r \text{ with } s \rightarrow t \succeq l \rightarrow r} \\
\text{Extend:} & (E; R) & \vdash & (E; R \cup \{s \rightarrow t\}) & \text{if } s \rightarrow t \in \mathbf{ex}_S(R) \\
\text{Deduce:} & (E; R) & \vdash & (E \cup \{s = t\}; R) & \text{if } s = t \in \mathbf{cp}_S(R)
\end{array}
$$

As before, we write $(E; R) \vdash_{KB/S} (E'; R')$ if the latter may be obtained from the former by one application of a rule in KB/S. With this inference system, **delete** removes an equation between S-equivalent terms; **collapse** simplifies left-hand sides; **extend** adds extended rules; **deduce** generates extended critical pairs. Extended rewriting requires S-matching; S-completion requires S-unification to generate critical pairs and extended rules. The set S is unchanging throughout.

Definition 25. An *S-completion procedure* is any program that takes a finite set E_0 of equations, an S-unification procedure, and a reduction ordering $\succ$ that commutes over S, and uses the above rules KB/S to generate a sequence of inferences from $(E_0; \emptyset)$.

The most important case, in practice, is when S is a set of commutativity (C) or associativity-commutativity (AC) axioms for some binary function symbols [Lankford-Ballantyne, 1977b; Peterson-Stickel, 1981]. For the purposes of AC-completion, rules are commonly *flattened* by removing nested occurrences of associative-commutative symbols. An associative-commutative unification algorithm is employed—in place of the standard (syntactic) unification algorithm—to generate the critical pairs in $\mathbf{cp}_{AC}(R_i)$, and associative-commutative matching is used to apply rules. For each rule $f(s,t) \rightarrow r$ headed by an associative-commutative symbol f, an extended rule $f(s, f(t,z)) \rightarrow f(r,z)$ is added and flattened out to $f(s,t,z) \rightarrow f(r,z)$; extensions of AC-extended rules are redundant.

For example, consider the same set E_0 as in Figure 5, along with the following set S of AC axioms:

$$
\begin{array}{rcl}
x \cdot y & \leftrightarrow & y \cdot x \\
x \cdot (y \cdot z) & \leftrightarrow & (x \cdot y) \cdot z
\end{array}
$$

Extend uses associativity to create two extended rules, $(1 \cdot x) \cdot z \rightarrow x \cdot z$ and $x^- \cdot (x \cdot z) \rightarrow 1 \cdot z$, the first of which collapses away, and the second of which composes to yield an existing rule. **Deduce** generates $(x \cdot y)^- \cdot x \rightarrow y^-$ from an S-overlap of $x^- \cdot (x \cdot y) \rightarrow y$ on itself (at position 2). The resultant rule is extended to $(x \cdot y)^- \cdot (x \cdot z) \rightarrow y^- \cdot z$, which forms an S-critical pair with $x^- \cdot x \rightarrow 1$ and generates $x^- \cdot y^- \rightarrow (x \cdot y)^-$. Extending as necessary, cleaning up, and flattening products, the final result is:

$$
\begin{array}{rcl|rcl}
1^- & \rightarrow & 1 & x \cdot 1 & \rightarrow & x \\
x \cdot x^- & \rightarrow & 1 & x \cdot x^- \cdot z & \rightarrow & z \\
(x^-)^- & \rightarrow & x & (x \cdot y^-)^- & \rightarrow & x^- \cdot y \\
(x \cdot y)^- \cdot x & \rightarrow & y^- & (x \cdot y)^- \cdot x \cdot z & \rightarrow & y^- \cdot z \\
x^- \cdot y^- & \rightarrow & (y \cdot x)^- & x^- \cdot y^- \cdot z & \rightarrow & (y \cdot x)^- \cdot z
\end{array}
$$

A better choice of ordering, one that would make $(y \cdot x)^-$ greater than $x^- \cdot y^-$, would result in the following neater system G/AC for Abelian (commutative) groups:

$$
\begin{array}{rcl|rcl}
1^- & \rightarrow & 1 & x \cdot 1 & \rightarrow & x \\
x \cdot x^- & \rightarrow & 1 & x \cdot x^- \cdot z & \rightarrow & z \\
(x^-)^- & \rightarrow & x & (y \cdot x)^- & \rightarrow & x^- \cdot y^-
\end{array}
$$

A proof in $S \cup E \cup R$ is a sequence of S-steps, E-steps, and R-steps. Analogous to the standard case, a relation $\Rightarrow_{C/S}$ can be defined that describes the simplifying effect of S-completion at the proof level. Using the Extended Critical Pair Lemma, it can then be shown that a completion sequence is fair (with respect to $\Rightarrow_{C/S}$) if all persistent critical pairs are accounted for ($\mathbf{cp}_S(R_\infty)$ is a subset of the S-variants of $\cup E_i$), all persistent extended rules are accounted for ($\mathbf{ex}_S(R_\infty)$ is a subset of the S-variants of $\cup R_i$), and no equation persists (E_∞ is empty). With fairness, we get that an extended-rewrite proof $s \rightarrow^*_{S\backslash R_\infty} \circ \leftrightarrow^*_S \circ \leftarrow^*_{S\backslash R_\infty} t$ will eventually be generated if $s \leftrightarrow_{S \cup E_0} t$ [Jouannaud-Kirchner, 1986]. However, the limit $S\backslash R_\infty$ need not be reduced.

Additional aspects of completion modulo equational theories have been considered: [Huet, 1980] deals with the left-linear case; [Jouannaud-Kirchner, 1986] analyze exactly which critical pairs are necessary when some rules are left-linear and others are not; [Bachmair-Dershowitz, 1987a] take the inference rule approach and generalize previous results.

8.4 Ordered Completion

We have seen that completion can have any one of three outcomes: it may succeed in finding a decision procedure for validity after a finite number of steps; it may loop and generate more and more rules until—at some point—any particular valid equation has a rewrite proof; or it may abort with unorientable equations before finding any proof.

Since there are total reduction orderings for any set of ground terms (the lexicographic path ordering with total precedence is one such), completion of ground terms—given such an ordering—will not abort. Moreover, ground completion need never apply the **deduce** inference rule, since the **collapse** rule always applies to one of the rules contributing to a critical pair. And without **deduce**, completion will not loop. Thus, for any finite set of ground equations, completion is sure to generate a decision procedure [Lankford, 1975], which is not surprising, since all such theories are decidable [Ackermann, 1954]. In fact, various $O(n \lg n)$ congruence-closure algorithms exist for the purpose (e.g. [Downey-etal, 1980]; see also [Snyder, 1989]). More interesting are those cases where there are non-ground rules for which a canonical rewrite system is available, and all critical pairs between a ground rule and a non-ground one are either ground or simplify to a trivial rule. By supplying completion with a complete simplification ordering, these critical pairs can always be oriented. (The complete ordering must be compatible with the canonical system for the non-ground rules.) For example, AC-completion can be used in this way to generate decision procedures for finitely-presented Abelian groups starting from G/AC [Lankford-etal, 1984].

We now turn our attention to completion of ordered rewriting systems, and call the process "ordered" (or "unfailing") completion. Ordered completion either returns a (finite) ordered rewriting system in finite time, or else loops and generates an infinite system. With a finite system, validity can be decided by ordered rewriting; an infinite system can only serve as a semi-decision procedure. Since all orientable instances of equations are used to rewrite, there will be no need to explicitly distinguish between rewrite rules and other equations in the inference rules. Let $\succ$ be a reduction ordering that can be extended to a complete simplification ordering, and let $\trianglerighteq$ be the encompassment ordering. Consider the following set OC of inference rules, operating on set of equations E (cf. [Bachmair-etal, 1989]):

Delete:	$E \cup \{s = s\}$	$\vdash$	E	
Simplify:	$E \cup \{s = t\}$	$\vdash$	$E \cup \{s = u\}$	if $t \rightarrow_\succ u$ and $s \succ u$
Collapse:	$E \cup \{s = t\}$	$\vdash$	$E \cup \{s = u\}$	if $t \rightarrow_\succ u$ by $l = r$ and $t \trianglerighteq l$
Deduce:	E	$\vdash$	$E \cup \{s = t\}$	if $s = t \in \mathbf{cp}_\succ(E)$

We write $E \vdash_{OC} E'$ if the latter may be obtained from the former by one application of a rule in OC. With this inference system, **deduce** generates ordered critical pairs, and the other rules simplify them.

Definition 26. An *ordered completion procedure* is any program that takes a finite set E_0 of equations and a reduction ordering $\succ$ that can be extended to a complete simplification ordering, and uses the above rules OC to generate a sequence of inferences from E_0.

For example, consider the following axioms for entropic groupoids:

$$\begin{array}{rcl} (x \cdot y) \cdot x & = & x \\ (x \cdot y_1) \cdot (z \cdot y_2) & = & (x \cdot z) \cdot (y_1 \cdot y_2) \end{array}$$

The second equation is permutative and cannot be oriented by any reduction ordering. Completion will therefore fail. Ordered completion, on the other hand, yields the ground-convergent ordered-rewriting system shown in Section 7.4.

Analogous to the standard case, a relation $\Rightarrow_{OC}$ can be defined that describes the simplifying effect of ordered completion at the proof level. *Fairness* is defined accordingly. Using the Ordered Critical Pair Lemma, it can then be shown that a completion sequence is fair (with respect to $\Rightarrow_{OC}$) if all persistent critical pairs are accounted for, i.e. if $\mathbf{cp}_{\succ}(E_\infty)$ is a subset of $\cup E_i$. With fairness, we get that a rewrite proof between two ground terms s and t will eventually be generated iff $s \leftrightarrow^*_{E_0} t$ [Hsiang-Rusinowitch, 1987]. Thus, the word problem in arbitrary equational theories can always be semi-decided by ordered completion. (See [Boudet-etal, 1988] for an interesting application.)

It is not hard to see that OC can mimic KB for any given equational theory E and reduction ordering $\succ$ (not necessarily total on $\mathcal{G}$). The natural question is whether ordered completion must succeed in generating a canonical set of rules whenever one exists for the given reduction ordering $\succ$. The answer is affirmative [Bachmair-etal, 1989], provided $\succ$ can be extended to a complete reduction ordering. For example, if b and c are incomparable, ordered completion infers

$$(\{b=c\};\{f(b)\rightarrow a, d\rightarrow b, f(c)\rightarrow c\}) \underset{OC}{\vdash} (\{b=c, f(c)=a\};\{f(b)\rightarrow a, d\rightarrow b, f(c)\rightarrow c\})$$

With the recursive path ordering in which $f \succ c \succ a$, this sequence continues until success:

$$\overset{+}{\underset{OC}{\vdash}} (\{b=c, c=a\};\{f(b)\rightarrow a, d\rightarrow b, f(c)\rightarrow c, f(c)\rightarrow a\}) \overset{+}{\underset{OC}{\vdash}} (\emptyset;\{f(a)\rightarrow a, b\rightarrow a, c\rightarrow a, d\rightarrow a\})$$

Ordered completion can also be modified to act as a refutationally complete inference system for validity in equational theories. To prove $s = t$, its Skolemized negation $eq(s', t') = F$ is added to the initial set $E \cup \{eq(x, x) = T\}$ of equations. With a completable reduction ordering (the Skolem constants are added to the algebra, hence must be comparable), the outcome $T = F$ characterizes validity of $s = t$ in the theory of E [Hsiang-Rusinowitch, 1987].

8.5 Inductive Theorem Proving

An inductive theorem-proving method based on completion was first proposed in [Musser, 1980]. Recall from Section 3.2 that an inductive theorem $s =_{I(E)} t$ holds iff there is no equation $u = v$ between ground terms that follows from $E \cup \{s = t\}$, but not from E alone. Let H be a set of equational hypotheses. Given a ground-convergent system R for E, we aim to find a ground-convergent system R' for $E \cup H$ with the same ground normal forms. If R' is the result of a successful completion sequence starting from $(H; R)$ and using an ordering containing R, then, by the nature of completion, R' is convergent and every term reducible by R is reducible by R'. To check that every ground term reducible by R' is also R-reducible, it is necessary and sufficient that every left-hand side of R' be ground R-reducible. If R' passes this test, then the two systems have the same ground normal forms and $s =_{I(R)} t$ for each $s = t$ in H [Dershowitz, 1982b; Jouannaud-Kounalis, 1989]. If, on the other hand, $s \neq_{I(R)} t$, then fair completion will uncover an inconsistency, if it does not fail. This approach is also valid when extended-completion (or ordered-completion) is used [Goguen, 1980].

For example, let $\mathcal{F}$ be $\{\Lambda, push, alternate\}$ and R be the following canonical system for interleaving stacks:

$$\begin{array}{rcl} alternate(\Lambda, z) & \rightarrow & z \\ alternate(push(x, y), z) & \rightarrow & push(x, alternate(z, y)) \end{array}$$

and suppose we wish to prove that $alternate(y, \Lambda) = y$ is an inductive theorem. This equation can be oriented from left to right. Since the critical pairs $\Lambda = \Lambda$ and $push(x, y) = push(x, alternate(\Lambda, y))$ are provable by rewriting, completion ends with the system $R' = R \cup \{alternate(y, \Lambda) \rightarrow y\}$. Since the left-hand side $alternate(y, \Lambda)$ of the new rule is ground R-reducible, for all ground terms y in $\mathcal{G}(\{\Lambda, push, alternate\})$, the theorem is valid in $I(R)$. In more complicated cases, additional lemmata may be generated along the way.

Finding an R' that is complete for $H \cup R$ (as completion does) is actually much more than needed to preclude inconsistency. One need only show that some R' provides a valley proof for all ground consequences of $H \cup R$ (and that $R(\mathcal{G}) \subseteq R'(\mathcal{G})$). Thus, for each inference step $(H_i, R_i) \vdash_{OC} (H_{i+1}, R_{i+1})$, the equational hypotheses in H_i should be inductive consequences of $H_{i+1} \cup R_i$, and the rules in R_{i+1} should be inductive consequences of R_i [Bachmair, 1988]. Then, at each stage n, $H_0 \cup R_0$ follows from $H_n \cup R_0$; the original hypotheses H_0 are proved as soon as an empty H_n suffices. Assuming R_i is always the original R, it is enough if any ground cliff $v \longleftarrow_R u \leftrightarrow_{H_i} w$ can be reduced to a smaller proof in $H_{i+1} \cup R$ (smaller, in some well-founded sense), and no instance of a hypothesis in H_i is itself an inconsistency (that is, an equation between two distinct ground R-normal forms). Consequently, there is no need (though it may help) to generate critical pairs between two rules derived from H_0; it suffices to consider critical pairs obtained by narrowing with R at a set of "covering" positions in H_i [Fribourg, 1986]. When, and if, these pairs all simplify away, the inductive theorems are proven. On the other hand, if any of the hypotheses are false, a contradiction must eventually surface in a fair derivation. The use of critical pair criteria in this connection is explored in [Gobel, 1987; Kuchlin, 1989]. How to handle inductive equations that cannot be oriented is discussed in [Bachmair, 1989b; Jouannaud-Kounalis, 1989].

8.6 First-Order Theorem Proving

Rewriting techniques have been applied to first-order theorem proving in two ways. One approach is to use resolution for non-equality literals together with some kind of superposition of left-hand sides of equality literals within clauses [Lankford-Ballantyne, 1979; Peterson, 1983; Hsiang-Rusinowitch, 1986; Bachmair, 1989a].

An alternative approach [Hsiang-Dershowitz, 1983] is to use the Boolean ring system BA of Section 2 and treat logical connectives equationally. Let R be BA, S be AC for xor and and, and $E = \{s = T\}$, where s is a logically unsatisfiable Boolean formula. It follows from Herbrand's Theorem [Herbrand, 1930] that a finite conjunction of instances $s\sigma_i$ has normal form F under BA/AC. Thus, AC-completion, starting with R and E, will reduce the proof

$$T \underset{BA}{\overset{*}{\longleftarrow}} and(\ldots and(T, T), \ldots, T) \underset{E}{\overset{*}{\longleftrightarrow}} and(\ldots and(s\sigma_1, s\sigma_2), \ldots, s\sigma_n) \longrightarrow^*_{BA/AC} F$$

to a critical pair $T = F$, if it does not fail. As with resolution theorem-proving, one can prove validity of a closed formula in first-order predicate calculus by deriving a contradiction from its Skolemized negation. There are several problems with such an approach: (a) AC-unification is required to compute critical pairs; (b) failure is possible unless the more expensive ordered procedure is used; and (c) critical pairs with the distributivity axiom in BA are very costly. Hsiang [1985] showed that if each equation in E is of the form $xor(s, T) = F$, where s is the exclusive-or normal form of a clause, then only a subset of the critical pairs with BA must be computed. (See [Muller-Socher-Ambrosius, 1988] for clarifications regarding the need for "factoring" if terms are simplified via BA.) This approach allows the integration of convergent systems for relevant equational theories, when such are available. A different completion-like procedure for first-order theorem proving, incorporating simplification, has been proved correct by [Bachmair-Dershowitz, 1987b]. A first-order method, using Boolean rings and based on polynomial ideals, is [Kapur-Narendran, 1985].

8.7 Further Reading

A survey of completion and its manifold applications may be found in [Dershowitz, 1989]. For a book focusing on the abstract view of completion and its variants, see [Bachmair, 1989b]. The relationship between completion and algorithms for finding canonical bases of polynomial ideals is discussed in [Loos, 1981] and [Buchberger, 1987]. Reve [Lescanne, 1983], Formel [Fages, 1984], Kads [Stickel, 1986], and Rrl [Kapur-Zhang, 1988] are four current implementations of AC-completion. Rewrite-based decision procedures for semigroups, monoids, and groups are investigated in [Benninghofen-etal, 1987]; experiments with the completion of finitely-presented algebras are described in [Lankford-etal, 1984; LeChenadec, 1985]; some new classes of decidable monadic word problems were found in [Pedersen, 1989]. The use of completion and its relation to Dehn's method for deciding word problems and small cancellation theory is explored in [LeChenadec, 1987].

Early forerunners of ordered completion were [Lankford, 1975; Brown, 1975]. Ordered completion modulo a congruence has been implemented by [Anantharaman-etal, 1989]. One of the first implementations of inductive theorem proving by completion was [Huet-Hullot, 1980]. A recent book on using rewrite techniques for theorem proving in first-order predicate calculus with equality is [Rusinowitch, 1989].

9 EXTENSIONS

In this section, we briefly consider four variations of the rewriting theme: "order-sorted rewriting," "conditional rewriting," "priority rewriting," and "graph rewriting."

9.1 Order-Sorted Rewriting

In ordered rewriting, replacement is constrained to make terms smaller in a given ordering; more syntactic means of limiting rewriting are obtained by taxonomies of term types. For example, some data may be of type Boolean, some may represent natural numbers, others, stacks of reals. The appropriate semantic notion in this case is the many-sorted (heterogeneous) algebra. Under reasonable assumptions, virtually everything we have said extends to the multisorted case. Sorted (i.e. typed) rewriting has been dealt with, for example, in [Huet-Oppen, 1980; Goguen-Meseguer, 1985].

Order-sorted algebras, introduced in [Goguen, 1978] and developed in [Gogolla, 1983; Dick-Cunningham, 1985; Goguen-etal, 1985; Goguen-Meseguer, 1987; Smolka-etal, 1989], and others, extend sorted algebras by imposing a partial ordering on the sorts, intended to capture the subset relation between them. For example, one can distinguish between stacks, in general, and non-empty stacks by declaring $NonEmptyStack$ to be a subsort of $Stack$ and specifying their operations to have the following types:

$$\begin{array}{rcrcl}
\Lambda & : & & \rightarrow & Stack \\
push & : & Nat \times Stack & \rightarrow & NonEmptyStack \\
\hline
top & : & NonEmptyStack & \rightarrow & Nat \\
pop & : & NonEmptyStack & \rightarrow & Stack
\end{array}$$

The main advantage is that definitions can be sufficiently complete without introducing error elements for functions applied outside their intended domains (like $top(\Lambda)$).

Free algebras can be constructed in the order-sorted case [Goguen-Meseguer, 1987], but the algebra obtained is, in general, an amalgamation of the term algebra. To avoid this complication, a syntactic "regularity" condition (namely, that every term has a least sort) may be imposed on the signature [Goguen-Meseguer, 1987]. Subsorts also require run-time checking for syntactic validity. For example, the term $pop(push(e, push(e, \Lambda)))$ has sort $Stack$, so its top cannot be taken, yet that term is equal to the nonempty stack $push(e, \Lambda)$. Fortunately, the run-time checks require no additional overhead [Goguen-etal, 1985].

Deduction in order-sorted algebras also presents some difficulties. For example, suppose that a sort S' contains two constants a and b, and S is the supersort $S' \cup \{c\}$. Let E consist of two identities: $a = c$ and $b = c$, and consider the equation $f(a) = f(b)$, where f is of type $S' \rightarrow S$. Clearly, the equation holds in all models, but this cannot be shown by replacement of equals, since $f(c)$ is not well-formed. Additional problems are caused when a sort is allowed to be empty, as pointed out in [Huet-Oppen, 1980]. A sound and complete set of inference rules for order-sorted equality is given in [Goguen-Meseguer, 1987].

For order-sorted rewriting, confluence does not imply that every theorem has a rewrite proof. Consider the same example as in the previous paragraph. The system $\{a \rightarrow c, b \rightarrow c\}$ is confluent, but there is no rewrite proof for $a = b$, since a term is not rewritten outside its sort. One way to preclude such anomalies is to insist that rules be "sort decreasing," i.e. that any instance of a right-hand side have a sort no larger than that of the corresponding left-hand side [Goguen-etal, 1985]. Order-sorted unification is investigated in [Walther, 1988; Meseguer-etal, 1989] and order-sorted completion, in [Kirchner, 1987].

9.2 Conditional Rewriting

Another way to restrict applicability of equations is to add enabling conditions. A *conditional equation* is an equational implication $u_1 = v_1 \wedge \cdots \wedge u_n = v_n \Rightarrow s = t$, for $n \geq 0$ (that is, a universal Horn clause with equality literals only). An example of a conditional equation with only one premiss is $empty?(x) = no \Rightarrow push(top(x), pop(x)) = x$. Initial algebras exist for classes of algebras presented by conditional equations. A *conditional rule* is an equational implication in which the equation in the conclusion is oriented, for which we use the format $u_1 = v_1 \wedge \cdots \wedge u_n = v_n \mid l \rightarrow r$. The following is an example of a system of both conditional and unconditional rules:

$$\begin{array}{rcrcl} & & top(push(x,y)) & \rightarrow & x \\ & & pop(push(x,y)) & \rightarrow & y \\ & & empty?(\Lambda) & \rightarrow & yes \\ & & empty?(push(x,y)) & \rightarrow & no \\ empty?(x) = no & \mid & push(top(x), pop(x)) & \rightarrow & x \end{array}$$

To give operational semantics to conditional systems, the conditions under which a rewrite may be performed need to be made precise. In the above example, is a term $push(top(s), pop(s))$ rewritten whenever the subterm s is such that the condition $empty?(s) = no$ can be proved, whenever a rewrite proof exists, or only when $empty?(s)$ rewrites to no? The ramifications of various choices are discussed in [Brand-etal, 1978; Bergstra-Klop, 1986; Dershowitz-etal, 1988]. The most popular convention is that each condition admit a rewrite proof; in other words, for a given system R, a rule $u_1 = v_1 \wedge \cdots \wedge u_n = v_n \mid l \rightarrow r$ is applied to a term $t[l\sigma]$ if $u_i\sigma \rightarrow_R^* \circ \leftarrow_R^* v_i\sigma$, for each condition $u_i = v_i$, in which case $t[l\sigma] \rightarrow_R t[r\sigma]$. Recent proposals for logic programming languages, incorporating equality, have been based on conditional rewriting and narrowing (e.g. [Dershowitz-Plaisted, 1985; Goguen-Meseguer, 1986]; see [Reddy, 1986].

For the above recursive definition of $\rightarrow_R$ to yield a decidable relation, restrictions must be made on the rules. The most general well-behaved proposal are *decreasing* systems, terminating systems with conditions that are smaller (in a well-defined sense) than left-hand sides [Dershowitz-etal, 1988]; hence, recursively evaluating the conditions always terminates. More precisely, a rule $u_1 = v_1 \wedge \cdots \wedge u_n = v_n \mid l \rightarrow r$ is decreasing if there exists a well-founded extension $\succ$ of the proper subterm ordering such that $\succ$ contains $\rightarrow_R$ and $l\sigma \succ u_i\sigma, v_i\sigma$ for $i = 1, \ldots, n$. Decreasing systems generalize the concept of "hierarchy" in the work of [Remy, 1982], and are slightly more general than the systems considered in [Kaplan, 1987; Jouannaud-Waldmann, 1986]; they have been extended in [Dershowitz-Okada, 1988b] to cover systems (important in logic programming) with variables in conditions that do not also appear in the left-hand side, e.g. $g(x) = z \mid f(x) \rightarrow h(z)$.

The rewrite relation $\rightarrow_R$ for decreasing systems is terminating, and—when there are only a finite number of rules—is decidable. For confluence, a suitable notion of "conditional critical pair," which is just the conditional equation derived from overlapping left-hand sides, is defined. The Critical Pair Lemma holds for decreasing systems R [Kaplan, 1987]: a decreasing system R is locally confluent (and hence convergent) iff there is a rewrite proof $s\sigma \rightarrow_R^* \circ \leftarrow_R^* t\sigma$ for each critical pair $u_1 = v_1 \wedge \cdots \wedge u_n = v_n \Rightarrow s = t$ and substitution σ such that $u_i\sigma = R\ v_i\sigma$ for $i = 1, \ldots, n$. Nevertheless, confluence is only semi-decidable, on account of the semi-decidability of satisfiability of the conditions $u_i = v_i$. For non-decreasing systems, even ones for which the rewrite relation is terminating, the rewrite relation may be undecidable [Kaplan, 1987], and the Critical Pair Lemma does not hold, though terminating systems having no critical pairs are confluent (see [Dershowitz-etal, 1987a]). To handle more general systems of conditional rules, rules must be overlapped on conditions, extending ordered completion to what has been called "oriented paramodulation" [Ganzinger, 1987; Dershowitz, 1988; Kounalis-Rusinowitch, 1988].

9.3 Priority Rewriting

In priority rewriting, the choice among several possible redexes is constrained to meet, *a priori*, given priorities on the rules. Priorities, then, are just a partial ordering of rules. The original Markov algorithms were priority string rewriting systems, in which the written order of the rules determined their priority. The following definition of subtraction of natural numbers and divisibility of integers by naturals illustrates the conciseness made possible by using subsorts and priorities:

$$
\begin{array}{lrcl}
 & Zero & = & \{0\} \\
 & Pos & = & succ(Zero) \cup succ(Pos) \\
 & Nat & = & Zero \cup Pos \\
 & Neg & = & neg(Pos) \\
 & Int & = & Nat \cup Neg \\
\hline
-: & Nat \times Nat & \rightarrow & Int \\
|: & Nat \times Int & \rightarrow & \{T, F, error\} \\
\hline
 & x - 0 & \rightarrow & x \\
 & 0 - y & \rightarrow & neg(y) \\
 & succ(x) - succ(y) & \rightarrow & x - y \\
 & 0|x & \rightarrow & error \\
 & z|x : Neg & \rightarrow & F \\
 & z|0 & \rightarrow & T \\
 & z|x & \rightarrow & z|(x - z)
\end{array}
$$

Note how priority systems (with a total priority ordering) can have no "critical" overlaps between two left-hand sides at the top. Priority term-rewriting systems were first formally studied in [Baeten-etal, 1984]. Their definition is subtle, since an outermost redex is rewritten only if no possible derivation of its proper subterms enables a higher-priority rule at the outermost position. A condition is given under which this definition of rewriting is computable. Priority systems cannot, in general, be expressed as term-rewriting systems.

9.4 Graph Rewriting

Rewriting has also been generalized to apply to graphs, instead of terms. In fact, the idea of rewriting all kinds of objects is already in [Thue, 1914]. In graph rewriting, subgraphs are replaced according to rules, containing variables, themselves referring to subgraphs. For example, a rule $f(g(x)) \rightarrow h(x)$, when applied to a directed acyclic graph $k(g(a), f(g(a)))$, where $g(a)$ is shared by k's two subterms, should rewrite the graph to $k(g(a), h(a))$, with the a still shared. Unlike trees, graphs do not have a simple structure lending itself to inductive definitions and proofs, for which reason, the graph-rewriting definitions, as introduced in [Ehrig, 1977] and simplified in [Raoult, 1984], have a global flavor. A categorical framework is used to precisely define matching and replacement; a rewriting is then a pushout in a suitable category. Though the categorical apparatus leads to apparently complicated definitions, many proofs, e.g. the Critical Pair Lemma, become nothing more than commutativity of diagrams.

A completely different approach to graph rewriting is taken in [Bauderon-Courcelle, 1987], where finite graphs are treated as algebraic expressions. Finitely-oriented labeled hypergraphs are considered as a set of hyperedges glued together by means of vertices. This generalizes the situation of words, with hyperedge labels as the constants, gluing for concatenation, and a set S of equational laws defining equivalence of expressions (instead of associativity). Class-rewriting in R/S is, then, tantamount to graph-rewriting with a system R.

ACKNOWLEDGEMENTS

We thank Leo Bachmair, Hubert Comon, Kokichi Futatsugi, Steve Garland, Rob Hasker, Jieh Hsiang, Gérard Huet, Stéphane Kaplan, Deepak Kapur, Pierre Lescanne, Joseph Loyall, George McNulty, José Meseguer, Mike Mitchell, Paul Purdom, and Hantao Zhang for their help with aspects of this survey. It goes without saying that our work in this field would not have been possible without the collaboration of many other colleagues and students over the years.

The first author's research was supported in part by the National Science Foundation under Grant DCR 85-13417, by the Center for Advanced Study of the University of Illinois at Urbana-Champaign, and by Hebrew University at Jerusalem. The second author's research was supported in part by the Greco Programmation du Centre National de la Recherche Scientifique, the ESPRIT project METEOR, and a CNRS/NSF grant.

References

[Ackermann, 1954] W. Ackermann, Solvable Cases of the Decision Problem, North-Holland, Amsterdam (1954).

[Anantharaman-etal, 1989] S. Anantharaman, J. Hsiang and J. Mzali, SbReve2: A term rewriting laboratory with (AC-)unfailing completion, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, pp. 533-537, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin (April 1989).

[Ardis, 1980] M. A. Ardis, Data abstraction transformations, Report TR-925, Ph.D. thesis, Department of Computer Science, University of Maryland. (August 1980).

[Avenhaus, 1985] J. Avenhaus, On the termination of the Knuth-Bendix completion algorithm, Report 120/84, Universität Kaiserslautern, Kaiserslautern, West Germany (1985).

[Baader, 1986] F. Baader, Unification in idempotent semigroups is of type zero, *J. Automated Reasoning* **2** (3) (1986).

[Bachmair-Dershowitz, 1986] L. Bachmair and N. Dershowitz, Commutation, transformation, and termination, *Proc. of the Eighth International Conference on Automated Deduction*, J. H. Siekmann, ed., Oxford, England, pp. 5-20. Available as Vol. 230, *Lecture Notes in Computer Science*, Springer, Berlin (July 1986).

[Bachmair-Dershowitz, 1987a] L. Bachmair and N. Dershowitz, Completion for rewriting modulo a congruence, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 192-203. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin; revised version to appear in *Theoretical Computer Science* [1988] (May 1987).

[Bachmair-Dershowitz, 1987b] L. Bachmair and N. Dershowitz, Inference rules for rewrite-based first-order theorem proving, *Proc. of the Second Symposium on Logic in Computer Science*, Ithaca, NY, pp. 331-337 (June 1987).

[Bachmair-Dershowitz, 1988] L. Bachmair and N. Dershowitz, Critical pair criteria for completion, *J. of Symbolic Computation* **6** (1), pp. 1-18 (1988).

[Bachmair-Plaisted, 1985] L. Bachmair and D. A. Plaisted, Associative path ordering, *J. of Symbolic Computation* **1** (4), pp. 329-349 (December 1985).

[Bachmair, 1988] L. Bachmair, Proof by consistency in equational theories, *Proc. of the Third IEEE Symposium on Logic in Computer Science*, Edinburgh, Scotland, pp. 228-233 (July 1988).

[Bachmair, 1989a] L. Bachmair, Proof normalization for resolution and paramodulation, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin, pp. 15-28 (April 1989).

[Bachmair, 1989b] L. Bachmair, Canonical Equational Proofs, Pitman-Wiley, London. To appear (1989).

[Bachmair-etal, 1986] L. Bachmair, N. Dershowitz and J. Hsiang, Orderings for equational proofs, *Proc. of the Symposium on Logic in Computer Science*, Cambridge, MA, pp. 346-357 (June 1986).

[Bachmair-etal, 1989] L. Bachmair, N. Dershowitz and D. A. Plaisted, Completion without failure, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 1-30 (1989).

[Baeten-etal, 1984] J. C. M. Baeten, J. A. Bergstra and J. W. Klop, Priority rewrite systems, Report CS-R8407, Math Centrum, Amsterdam (1984).

[Barendregt, 1984] H. P. Barendregt, The Lambda Calculus, its Syntax and Semantics, North-Holland, Amsterdam. Second edition (1984).

[Barendregt, 1989] H. Barendregt, Functional programming and lambda calculus, in: *Handbook of Theoretical Computer Science*, J. van Leeuwen, ed., North-Holland, Amsterdam (1989).

[Bauderon-Courcelle, 1987] M. Bauderon and B. Courcelle, Graph expressions and graph rewritings, *Mathematical Systems Theory* **20**, pp. 83-127 (1987).

[Baxter, 1976] L. D. Baxter, A practically linear unification algorithm, Report CS-76-13, University of Waterloo, Waterloo, Canada (1976).

[Bellegarde-Lescanne, 1987] F. Bellegarde and P. Lescanne, Transformation orderings, *Proc. of the Twelfth Colloquium on Trees in Algebra and Programming* (1987).

[BenCherifa-Lescanne, 1987] A. Ben Cherifa and P. Lescanne, Termination of rewriting systems by polynomial interpretations and its implementation, *Science of Computer Programming* **9** (2), pp. 137-159 (October 1987).

[Benninghofen-etal, 1987] B. Benninghofen, S. Kemmerich and M. M. Richter, Systems of Reductions, *Lecture Notes in Computer Science* **277**, Springer, Berlin (1987).

[Bergstra-Klop, 1986] J. A. Bergstra and J. W. Klop, Conditional rewrite rules: Confluency and termination, *J. of Computer and System Sciences* **32**, pp. 323-362 (1986).

[Birkhoff, 1935] G. Birkhoff, On the structure of abstract algebras, *Proc. of the Cambridge Philosophical Society* **31**, pp. 433-454 (1935).

[Birkhoff, 1948] G. Birkhoff, Lattice Theory, American Mathematical Society, New York. Revised edition (1948).

[Bledsoe, 1977] W. Bledsoe, Non-resolution theorem proving, *Artificial Intelligence* **9**, pp. 1-35 (1977).

[Bockmayr, 1987] A. Bockmayr, A note on a canonical theory with undecidable unification and matching problem, *J. of Automated Reasoning* **3**, pp. 379-381 (1987).

[Book, 1987] R. V. Book, Thue systems as rewriting systems, *J. of Symbolic Computation* **3** (1&2), pp. 39-68 (February/April 1987).

[Book-etal, 1981] R. V. Book, M. Jantzen and C. Wrathan, Monadic Thue systems, *Theoretical Computer Science* **19** (3), pp. 231-251 (1981).

[Bloom-Tindell, 1983] S. L. BLOOM and R. TINDELL, Varieties of 'if-then-else', *SIAM J. on Computing* **12** (4), pp. 677-707 (November 1983).

[Boudet, 1989] A. BOUDET, AC-unification is easy, Report, Laboratoire de Recherche en Informatique, Universite Paris-Sud, Orsay, France (April 1989).

[Boudet-etal, 1988] A. BOUDET, J.-P. JOUANNAUD and M. SCHMIDT-SCHAUSS, Unification in free extensions of Boolean rings and Abelian groups, *Proc. of the Third Symposium on Logic in Computer Science*, Edinburgh, Scotland, pp. 121-130. To appear in *J. of Symbolic Computation* (July 1988).

[Brand-etal, 1978] D. BRAND, J. A. DARRINGER and W. J. JOYNER, JR., Completeness of conditional reductions, Report RC 7404, IBM Thomas J. Watson Research Center, Yorktown Heights, NY (December 1978).

[Brown, 1975] T. C. BROWN, JR., A structured design-method for specialized proof procedures, Ph.D. thesis, California Institute of Technology, Pasadena, CA (1975).

[Burckert-etal, 1987] H.-J. J. BÜRCKERT, A. HEROLD and M. SCHMIDT-SCHAUSS, On equational theories, unification and decidability, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 204-215. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin (May 1987).

[Buchberger-Loos, 1982] B. BUCHBERGER and R. LOOS, Algebraic simplification, in: *Computer Algebra*, Springer, Berlin, pp. 11-43 (1982).

[Buchberger, 1987] B. BUCHBERGER, History and basic features of the critical-pair/completion procedure, *J. of Symbolic Computation* **3** (1&2), pp. 3-38 (February/April 1987).

[Butler-Lankford, 1980] G. BUTLER and D. S. LANKFORD, Experiments with computer implementations of procedures which often derive decision algorithms for the word problem in abstract algebras, Memo MTP-7, Department of Mathematics, Louisiana Tech. University, Ruston, LA (August 1980).

[Choppy-etal, 1987] C. CHOPPY, S. KAPLAN and M. SORIA, Algorithmic complexity of term rewriting systems, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 256-285. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin (May 1987).

[Church-Rosser, 1936] A. CHURCH and J. B. ROSSER, Some properties of conversion, *Transactions of the American Mathematical Society* **39**, pp. 472-482 (1936).

[Cohn, 1981] P. M. COHN, Universal Algebra, D. Reidel, Dordrecht, Holland. Second edition (1981).

[Collins, 1975] G. COLLINS, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Proc. Second GI Conference on Automata Theory and Formal Languages*, pp. 134-183. Available as Vol. 33, *Lecture Notes in Computer Science*, Springer, Berlin (1975).

[Colmerauer, 1984] A. COLMERAUER, Equations and inequations on finite and infinite trees, *Proc. of the Second International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, pp. 85-99 (1984).

[Comon-Lescanne, 1989] H. COMON and P. LESCANNE, Equational problems and disunification, *J. of Symbolic Computation*. To appear (1989).

[Comon, 1989] H. COMON, Disunification and inductive proofs, *J. of Computer Systems and Sciences*. To appear (1989).

[Curry-Feys, 1958] H. B. CURRY and R. FEYS, Combinatory Logic, **1**, North-Holland, Amsterdam (1958).

[Dauchet, 1989] M. DAUCHET, Simulation of Turing machines by a left-linear rewrite rule, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin, pp. 109-120 (April 1989).

[Dauchet-etal, 1987] M. DAUCHET, S. TISON, T. HEUILLARD and P. LESCANNE, Decidability of the confluence of ground term rewriting systems, *Proc. of the Second Symposium on Logic in Computer Science*, Ithaca, NY, pp. 353-359 (June 1987).

[Davis-Hersh, 1973] M. DAVIS and R. HERSH, Hilbert's 10th problem, *Scientific American* **229** (5), pp. 84-91 (November 1973).

[Dehn, 1911] M. DEHN, Uber unendliche diskontinuierliche Gruppen, *Mathematische Ann.* **71**, pp. 116-144 (1911).

[Dershowitz-Manna, 1979] N. DERSHOWITZ and Z. MANNA, Proving termination with multiset orderings, *Communications of the ACM* **22** (8), pp. 465-476 (August 1979).

[Dershowitz-Okada, 1988a] N. DERSHOWITZ and M. OKADA, Proof-theoretic techniques and the theory of rewriting, *Proc. of the Third Symposium on Logic in Computer Science*, Edinburgh, Scotland, pp. 104-111 (July 1988).

[Dershowitz-Okada, 1988b] N. DERSHOWITZ and M. OKADA, Conditional equational programming and the theory of conditional term rewriting, *Proc. of the International Conference on Fifth Generation Computer Systems*, Tokyo, Japan (November 1988).

[Dershowitz-Plaisted, 1985] N. DERSHOWITZ and D. A. PLAISTED, Logic programming *cum* applicative programming, *Proc. of the IEEE Symposium on Logic Programming*, Boston, MA, pp. 54-66 (July 1985).

[Dershowitz, 1979] N. DERSHOWITZ, A note on simplification orderings, *Information Processing Letters* **9** (5), pp. 212-215 (November 1979).

[Dershowitz, 1982a] N. DERSHOWITZ, Orderings for term-rewriting systems, *Theoretical Computer Science* **17** (3), pp. 279-301 (March 1982).

[Dershowitz, 1982b] N. DERSHOWITZ, Applications of the Knuth-Bendix completion procedure, *Proc. of the Seminaire d'Informatique Theorique*, Paris, France, pp. 95-111 (December 1982).

[Dershowitz, 1984] N. DERSHOWITZ, Equations as programming language, *Proc. of the Fourth Jerusalem Conference on Information Technology*, IEEE Computer Society, pp. 114-123, pp. 114-124 (May 1984).

[Dershowitz, 1985] N. DERSHOWITZ, Computing with rewrite systems, *Information and Control* **64** (2/3), pp. 122-157 (May/June 1985).

[Dershowitz, 1987] N. DERSHOWITZ, Termination of rewriting, *J. of Symbolic Computation* **3** (1&2), pp. 69-115. Corrigendum [December 1987], Vol. 4, No. 3, pp. 409-410 (February/April 1987).

[Dershowitz, 1988] N. DERSHOWITZ, Proving equational Horn clauses, . Submitted (1988).

[Dershowitz, 1989] N. DERSHOWITZ, Completion and its applications, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 31-86 (1989).

[Dershowitz-etal, 1987a] N. DERSHOWITZ, M. OKADA and G. SIVAKUMAR, Confluence of conditional rewrite systems, *Proc. of the First International Workshop on Conditional Term Rewriting Systems*, S. Kaplan, J.-P. Jouannaud, ed., Orsay, France. Available as Vol. 308, *Lecture Notes in Computer Science*, Springer, Berlin [1988], pp. 31-44 (July 1987).

[Dershowitz-etal, 1987b] N. Dershowitz and G. Sivakumar, Solving goals in equational languages, *Proc. of the First International Workshop on Conditional Term Rewriting Systems*, S. Kaplan, J.-P. Jouannaud, ed., Orsay, France. Available as Vol. 308, *Lecture Notes in Computer Science*, Springer, Berlin [1988], pp. 45-55 (July 1987).

[Dershowitz-etal, 1988] N. Dershowitz, L. Marcus and A. Tarlecki, Existence, uniqueness, and construction of rewrite systems, *SIAM J. on Computing* **17** (4), pp. 629-639 (August 1988).

[Dershowitz-etal, 1988] N. Dershowitz, M. Okada and G. Sivakumar, Canonical conditional rewrite systems, *Proc. of the Ninth Conference on Automated Deduction*, Argonne, IL, pp. 538-549. Available as Vol. 310, *Lecture Notes in Computer Science*, Springer, Berlin (May 1988).

[Dick-Cunningham, 1985] R. J. Cunningham and A. J. J. Dick, Rewrite systems on a lattice of types, *Acta Informatica* **22**, pp. 149-169 (1985).

[Downey-etal, 1980] P. J. Downey, R. Sethi and R. E. Tarjan, Variations on the common subexpressions problem, *J. of the Association for Computing Machinery* **27** (4), pp. 758-771 (1980).

[Ehrenfeucht-etal, 1983] A. Ehrenfeucht, D. Haussler and G. Rozenberg, On regularity of context-free languages, *Theoretical Computer Science* **27** (3), pp. 311-332 (December 1983).

[Ehrig, 1977] H. Ehrig, Introduction to the algebraic theory of graph grammars, *Proc. of the International Conferenece on the Fundamentals of Complexity Theory*, Poznán-Kórnik, Poland, pp. 245-255. Available as Vol. 56, *Lecture Notes in Computer Science*, Springer, Berlin (1977).

[Evans, 1951] T. Evans, On multiplicative systems defined by generators and relations, I, *Proc. of the Cambridge Philosophical Society* **47**, pp. 637-649 (1951).

[Fages-Huet, 1983] F. Fages and G. Huet, Unification and matching in equational theories, *Proc. of the Eighth Colloquium on Trees in Algebra and Programming*, l'Aquilla, Italy, pp. 205-220. Available as Vol. 159, *Lecture Notes in Computer Science*, Springer, Berlin (1983).

[Fages, 1984] F. Fages, Le système KB: manuel de référence: présentation et bibliographie, mise en oeuvre, Report R. G. 10.84, Greco de Programmation, Bordeaux, France (1984).

[Fages, 1987] F. Fages, Associative-commutative unification, *J. of Symbolic Computation* **3** (3), pp. 257-275 (June 1987).

[Fay, 1979] M. Fay, First-order unification in an equational theory, *Proc. of the Fourth Workshop on Automated Deduction*, Austin, TX, pp. 161-167 (February 1979).

[Feferman, 1968] S. Feferman, Systems of predicative analysis II: Representation of ordinals, *J. of Symbolic Logic* **33** (2), pp. 193-220 (June 1968).

[Filman, 1978] R. E. Filman, Personal communication (1978).

[Freese-etal, 1989] R. Freese, R. McKenzie, G. F. McNulty and W. Taylor, Algebras, Lattices, Varieties, **II**, Wadsworth, Monterey, CA. To appear (1989).

[Fribourg, 1986] L. Fribourg, A strong restriction of the inductive completion procedure, *Proc. of the Thirteenth EATCS International Conference on Automata, Languages and Programming*, L. Kott, ed., Rennes, France, pp. 105-115. Available as Vol. 226, *Lecture Notes in Computer Science*, Springer, Berlin; to appear in *J. Symbolic Computation* (July 1986).

[Futatsugi-etal, 1985] K. Futatsugi, J. A. Goguen, J.-P. Jouannaud and J. Meseguer, Principles of OBJ2, *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, New Orleans, LA, pp. 52-66 (January 1985).

[Gallier-Snyder, 1987] J. H. GALLIER and W. SNYDER, A general complete E-unification procedure, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 192-203. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin (May 1987).

[Ganzinger, 1987] H. GANZINGER, A completion procedure for conditional equations, *Proc. of the First International Workshop on Conditional Term Rewriting Systems*, S. Kaplan, J.-P. Jouannaud, ed., Orsay, France. Available as Vol. 308, *Lecture Notes in Computer Science*, Springer, Berlin [1988], pp. 62-83 (July 1987).

[Gardner, 1983] M. GARDNER, Mathematical games: Tasks you cannot help finishing no matter how hard you try to block finishing them, *Scientific American* **24** (2), pp. 12-21 (August 1983).

[Gobel, 1987] R. GÖBEL, Ground confluence, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 156-167. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin (May 1987).

[Gogolla, 1983] M. GOGOLLA, Algebraic specifications with partially ordered sets and declarations, *Forschungsbericht Informatik 169*, Universität Dortmund, Dortmund, West Germany (1983).

[Goguen-Meseguer, 1985] J. A. GOGUEN and J. MESEGUER, Completeness of many sorted equational deduction, *Houston J. of Mathematics* **11** (3), pp. 307-334 (1985).

[Goguen-Meseguer, 1986] J. A. GOGUEN and J. MESEGUER EQLOG: Equality, types, and generic modules for logic programming, in: *Logic Programming: Functions, Relations, and Equations*, D. DeGroot, G. Lindstrom, ed., Prentice-Hall, Englewood Cliffs, NJ, pp. 295-363 (1986).

[Goguen-Meseguer, 1987] J. A. GOGUEN and J. MESEGUER, Order-sorted algebra I: Partial and overloaded operators, errors and inheritance, Unpublished report, SRI International, Menlo Park, CA (1987).

[Goguen-Tardo, 1979] J. A. GOGUEN and J. J. TARDO, An introduction to OBJ: A language for writing and testing formal algebraic specifications, *Proc. of the Specification of Reliable Software Conference*, pp. 170-189 (April 1979).

[Goguen, 1978] J. A. GOGUEN, Exception and error sorts, coercion and overloading operators, Research Report, Stanford Research Institute (1978).

[Goguen, 1980] J. A. GOGUEN, How to prove algebraic inductive hypotheses without induction, with applications to the correctness of data type implementations, *Proc. of the Fifth International Conference on Automated Deduction*, Les Arcs, France, pp. 356-373. Available as Vol. 87, *Lecture Notes in Computer Science*, Springer, Berlin, W. Bibel, ed., R. Kowalski, ed. (July 1980).

[Goguen-etal, 1985] J. A. GOGUEN, J.-P. JOUANNAUD and J. MESEGUER, Operational semantics of order-sorted algebra, *Proc. of the EATCS International Conference on Automata, Languages and Programming*, pp. 221-231. Available as Vol. 194, *Lecture Notes in Computer Science*, Springer, Berlin (1985).

[Goldfarb, 1981] W. D. GOLDFARB, Note on the undecidability of the second-order unification problem, *Theoretical Computer Science*, pp. 225-230 (1981).

[Gorn, 1967] S. GORN, Explicit definitions and linguistic dominoes, in: *Systems and Computer Science*, J. Hart, S. Takasu, ed., University of Toronto Press, pp. 77-115 (1967).

[Gries, 1981] D. GRIES, The Science of Programming, Springer, New York (1981).

[Guessarian-Meseguer, 1987] I. GUESSARIAN and J. MESEGUER, On the axiomatization of 'if-then-else', *SIAM J. on Computing* **16** (2), pp. 332-357 (April 1987).

[Guttag, 1976] J. V. GUTTAG, Abstract data types and the development of data structures, *Communications of the ACM* **20** (6), pp. 396-404 (July 1976).

[Heilbrunner-Holldobler, 1987] S. HEILBRUNNER and S. HÖLLDOBLER, The undecidability of the unification and matching problem for canonical theories, *Acta Informatica* **24** (2), pp. 157-171 (April 1987).

[Henkin, 1977] L. HENKIN, The logic of equality, *American Mathematical Monthly* **84** (8), pp. 597-612 (October 1977).

[Herbrand, 1930] J. HERBRAND, Recherches sur la théorie de la démonstration, Thèse de doctorat, Université de Paris, Paris, France (1930).

[Herold-Siekmann, 1987] A. HEROLD and J. SIEKMANN, Unification in Abelian semigroups, *J. of Automated Reasoning* **3** (3), pp. 247-283 (1987).

[Higman-Neumann, 1952] G. HIGMAN and B. H. NEUMANN, Groups as groupoids with one law, *Publ. Math. Debrecen* **2**, pp. 215-221 (1952).

[Higman, 1952] G. HIGMAN, Ordering by divisibility in abstract algebras, *Proc. of the London Mathematical Society (3)* **2** (7), pp. 326-336 (September 1952).

[Hindley, 1964] J. R. HINDLEY, The Church-Rosser property and a result in combinatory logic, Ph.D. thesis, University of Newcastle-upon-Tyne (1964).

[Hoffmann-O'Donnell, 1979] C. M. HOFFMANN and M. J. O'DONNELL, Interpreter generation using tree pattern matching, *Proc. of the Sixth ACM Symposium on Principles of Programming Languages*, San Antonio, TX, pp. 169-179 (January 1979).

[Hsiang-Dershowitz, 1983] J. HSIANG and N. DERSHOWITZ, Rewrite methods for clausal and non-clausal theorem proving, *Proc. of the Tenth EATCS International Colloquium on Automata, Languages and Programming*, Barcelona, Spain, pp. 331-346. Available as Vol. 154, *Lecture Notes in Computer Science*, Springer, Berlin (July 1983).

[Hsiang-Jouannaud, 1988] J. HSIANG and J.-P. JOUANNAUD, Complete sets of inference rules for E-unification, *Proc. of the Second Unification Workshop*, Val d'Ajol, France, Kirchner, C., ed. Available CRIN Report, Universite de Nancy, France (May 1988).

[Hsiang-Rusinowitch, 1986] J. HSIANG and M. RUSINOWITCH, A new method for establishing refutational completeness in theorem proving, *Proc. of the Eighth International Conference on Automated Deduction*, J. H. Siekmann, ed., Oxford, England, pp. 141-152. Available as Vol. 230, *Lecture Notes in Computer Science*, Springer, Berlin (July 1986).

[Hsiang-Rusinowitch, 1987] J. HSIANG and M. RUSINOWITCH, On word problems in equational theories, *Proc. of the Fourteenth EATCS International Conference on Automata, Languages and Programming*, T. Ottmann, ed., Karlsruhe, West Germany, pp. 54-71 (July 1987).

[Hsiang, 1982] J. HSIANG, Topics in automated theorem proving and program generation, Report R-82-1113, Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana, IL (December 1982).

[Hsiang, 1985] J. HSIANG, Refutational theorem proving using term-rewriting systems, *Artificial Intelligence* **25**, pp. 255-300 (March 1985).

[Huet-Hullot, 1980] G. HUET and J.-M. HULLOT, Proofs by induction in equational theories with constructors, *Proc. of the Twenty-First Annual Symposium on Foundations of Computer Science*, Lake Placid, NY, pp. 96-107 (October 1980).

[Huet-Lankford, 1978] G. HUET and D. S. LANKFORD, On the uniform halting problem for term rewriting systems, Rapport laboria 283, Institut de Recherche en Informatique et en Automatique, Le Chesnay, France (March 1978).

[Huet-Oppen, 1980] G. HUET and D. C. OPPEN, Equations and rewrite rules: A survey, in: *Formal Language Theory: Perspectives and Open Problems*, R. Book, ed., Academic Press, New York, pp. 349-405 (1980).

[Huet, 1976] G. HUET, Résolution d'équations dans des langages d'ordre 1, 2, $\ldots\omega$, Thèse de doctorat, Université de Paris VII, Paris, France (1976).

[Huet, 1980a] G. HUET, Personal communication (July 1980).

[Huet, 1980] G. HUET, Confluent reductions: Abstract properties and applications to term rewriting systems, *J. of the Association for Computing Machinery* **27** (4), pp. 797-821 (October 1980).

[Huet, 1981] G. HUET, A complete proof of correctness of the Knuth-Bendix completion algorithm, *J. Computer and Systems Sciences* **23** (1), pp. 11-21 (1981).

[Huet, 1985] G. HUET, Cartesian closed categories and Lambda-calculus, *Proc. of the LITP Spring School on Combinators and Functional Programming Languages*, Val d'Ajol, France, pp. 123-135. Available as Vol. 242, *Lecture Notes in Computer Science*, Springer, Berlin (May 1985).

[Hullot, 1980] J.-M. HULLOT, Canonical forms and unification, *Proc. of the Fifth International Conference on Automated Deduction*, Les Arcs, France, pp. 318-334. Available as Vol. 87, *Lecture Notes in Computer Science*, Springer, Berlin, W. Bibel, ed., R. Kowalski, ed. (July 1980).

[Iturriaga, 1967] R. ITURRIAGA, Contributions to mechanical mathematics, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA (1967).

[Jouannaud-Kirchner, 1986] J.-P. JOUANNAUD and H. KIRCHNER, Completion of a set of rules modulo a set of equations, *SIAM J. on Computing* **15**, pp. 1155-1194 (November 1986).

[Jouannaud-Kounalis, 1989] J.-P. JOUANNAUD and E. KOUNALIS, Automatic proofs by induction in equational theories without constructors, *Information and Computation.* To appear (1989).

[Jouannaud-Lescanne, 1987] J.-P. JOUANNAUD and P. LESCANNE, Rewriting systems, *Technology and Science of Informatics* **6** (3) Wiley, pp. 181-199. Available in French as "La réécriture" *Technique et Science de l'Informatique* [1986], vol. 5, no. 6, pp. 433-452 (1987).

[Jouannaud-Munoz, 1984] J.-P. JOUANNAUD and M. MUÑOZ, Termination of a set of rules modulo a set of equations, *Proc. of the Seventh International Conference on Automated Deduction*, R. E. Shostak, ed., Napa, CA, pp. 175-193. Available as Vol. 170, *Lecture Notes in Computer Science*, Springer, Berlin (May 1984).

[Jouannaud-Waldmann, 1986] J.-P. JOUANNAUD and B. WALDMANN, Reductive conditional term rewriting systems, *Proc. of the Third IFIP Working Conference on Formal Description of Programming Concepts*, Ebberup, Denmark (1986).

[Jouannaud, 1983] J.-P. JOUANNAUD, Confluent and coherent sets of reductions with equations: Application to proofs in abstract data types, *Proc. of the Eighth Colloquium on Trees in Algebra and Programming*, G. Ausiello, M. Protasi, ed., pp. 269-283. Available as Vol. 59, *Lecture Notes in Computer Science*, Springer, Berlin (1983).

[Jouannaud-etal, 1983] J.-P. JOUANNAUD, C. KIRCHNER and H. KIRCHNER, Incremental construction of unification algorithms in equational theories, *Proc. of the Tenth EATCS International Colloquium on Automata, Languages and Programming*, Barcelona, Spain, pp. 361-373. Available as Vol. 154, *Lecture Notes in Computer Science*, Springer, Berlin (July 1983).

[Kamin-Levy, 1980] S. KAMIN and J.-J. LÉVY, Two generalizations of the recursive path ordering, Unpublished note, Department of Computer Science, University of Illinois, Urbana, IL (February 1980).

[Kaplan, 1987] S. KAPLAN, Simplifying conditional term rewriting systems: Unification, termination and confluence, *J. of Symbolic Computation* **4** (3), pp. 295-334 (December 1987).

[Kapur-Musser, 1987] D. KAPUR and D. R. MUSSER, Proof by consistency, *Artificial Intelligence* **31** (2), pp. 125-157 (February 1987).

[Kapur-Narendran, 1985] D. KAPUR and P. NARENDRAN, An equational approach to theorem proving in first-order predicate calculus, *Proc. of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1146-1153 (August 1985).

[Kapur-Narendran, 1986] D. KAPUR and P. NARENDRAN, NP-completeness of set unification and matching problems, *Proc. of the Eighth International Conference on Automated Deduction*, Oxford, England, J. H. Siekmann, ed. Available as Vol. 230 *Lecture Notes in Computer Science*, Springer, Berlin Springer, pp. 489-495 (July 1986).

[Kapur-Narendran, 1987] D. KAPUR and P. NARENDRAN, Matching, unification and complexity (A preliminary note), *SIGSAM Bulletin* **21** (4), pp. 6-9 (November 1987).

[Kapur-Zhang, 1988] D. KAPUR and H. ZHANG, An overview of Rewrite Rule Laboratory (RRL), *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, pp. 559-563, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin (April 1989).

[Kapur-etal, 1986] D. KAPUR, P. NARENDRAN and H. ZHANG, Proof by induction using test sets, *Proc. of the Eighth International Conference on Automated Deduction*, J. H. Siekmann, ed., Oxford, England, pp. 99-117. Available as Vol. 230, *Lecture Notes in Computer Science*, Springer, Berlin (July 1986).

[Kapur-etal, 1987] D. KAPUR, P. NARENDRAN and F. OTTO, On ground confluence of term rewriting systems, Technical Report, Dept. of Computer Science, State University of New York at Albany, Albany, NY. To appear in *Information and Computation* (March 1987).

[Kapur-etal, 1987] D. KAPUR, P. NARENDRAN, D. J. ROSENKRANTZ and H. ZHANG, Sufficient-completeness, quasi-reducibility and their complexity, Technical report, Department of Computer Science, State University of New York, Albany, NY (1987).

[Kapur-etal, 1987] D. KAPUR, P. NARENDRAN and H. ZHANG, On sufficient completeness and related properties of term rewriting systems, *Acta Informatica* **24** (4), pp. 395-415 (August 1987).

[Kapur-etal, 1988] D. KAPUR, D. R. MUSSER and P. NARENDRAN, Only prime superpositions need be considered for the Knuth-Bendix procedure, *J. of Symbolic Computation* **4**, pp. 19-36 (August 1988).

[Kirby-Paris, 1982] L. KIRBY and J. PARIS, Accessible independence results for Peano arithmetic, *Bulletin London Mathematical Society* **14**, pp. 285-293 (1982).

[Kirchner, 1985] C. KIRCHNER, Methodes et outils de conception systematique d'algorithmes d'unification dans les theories equationnelles, Thèse d'Etat, Université de Nancy, Nancy, France (June 1985).

[Kirchner, 1986] C. KIRCHNER, Computing unification algorithms, *Proc. of the First IEEE Symposium on Logic in Computer Science*, Cambridge, Massachussets, pp. 206-216 (June 1986).

[Kirchner, 1987] C. Kirchner, *Proc. of the Colloquium on Trees in Algebra and Programming*, Order-sorted equational unification (1987).

[Kirchner, 1989] C. Kirchner, From unification in combination of equational theories to a new AC-unification algorithm, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 171-210 (1989).

[Klop, 1980] J. W. Klop, Reduction cycles in combinatory logic, in: *To H. B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. P. Seldin, R. Hendley, ed., Academic Press, New York, pp. 193-214 (1980).

[Klop, 1987] J. W. Klop, Term rewriting systems: A tutorial, *Bulletin of the European Association for Theoretical Computer Science* **32**, pp. 143-183 (June 1987).

[Knight, 1989] K. Knight, Unification: A multidisciplinary survey, *Computing Surveys* **21** (1), pp. 93-124 (March 1989).

[Knuth-Bendix, 1970] D. E. Knuth and P. B. Bendix, Simple word problems in universal algebras, in: *Computational Problems in Abstract Algebra*, J. Leech, ed., Pergamon Press, Oxford, U. K., pp. 263-297. Reprinted [1983] in *Automation of Reasoning 2*, Springer, Berlin, pp. 342-376 (1970).

[Kounalis-Rusinowitch, 1988] E. Kounalis and M. Rusinowitch, On word problems in Horn theories, *Proc. of the Ninth International Conference on Automated Deduction*, E. Lusk, R. Overbeek, ed., Argonne, Illinois. Available as Vol. 310, *Lecture Notes in Computer Science*, Springer, Berlin, pp. 527-537 (May 1988).

[Kounalis, 1985] E. Kounalis, Validation de spécifications algébriques par complétion inductive, Thèse d'Etat, Université de Nancy, Nancy, France (July 1985).

[Krishnamoorthy-Narendran, 1984] M. S. Krishnamoorthy and P. Narendran, A note on recursive path ordering, Unpublished note, General Electric Corporate Research and Development, Schenectady, NY (1984).

[Kruskal, 1960] J. B. Kruskal, Well-quasi-ordering, the Tree Theorem, and Vazsonyi's conjecture, *Transactions of the American Mathematical Society* **95**, pp. 210-225 (May 1960).

[Kruskal, 1972] J. B. Kruskal, The theory of well-quasi-ordering: A frequently discovered concept, *J. Combinatorial Theory Ser. A* **13** (3), pp. 297-305 (November 1972).

[Kuchlin, 1989] W. Küchlin, Inductive completion by ground proof transformation, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 211-244 (1989).

[Lankford-Ballantyne, 1977a] D. S. Lankford and A. M. Ballantyne, Decision procedures for simple equational theories with permutative axioms: complete sets of permutative reductions, ATP-37, Departments of Mathematics and Computer Sciences, University of Texas, Austin, TX (April 1977).

[Lankford-Ballantyne, 1977b] D. S. Lankford and A. M. Ballantyne, Decision procedures for simple equational theories with commutative-associative axioms: Complete sets of commutative-associative reductions, Memo ATP-39, Department of Mathematics and Computer Sciences, University of Texas, Austin, TX (August 1977).

[Lankford-Ballantyne, 1979] D. S. Lankford and A. M. Ballantyne, The refutation completeness of blocked permutative narrowing and resolution, *Proc. of the Fourth Workshop on Automated Deduction*, Austin, TX, pp. 53-59 (February 1979).

[Lankford-Ballantyne, 1983] D. S. Lankford and A. M. Ballantyne, On the uniqueness of term rewriting systems, Unpublished note, Department of Mathematics, Louisiana Tech. University, Ruston, LA (December 1983).

[Lankford, 1975] D. S. Lankford, Canonical inference, Memo ATP-32, Automatic Theorem Proving Project, University of Texas, Austin, TX (December 1975).

[Lankford, 1975] D. S. Lankford, Canonical algebraic simplification in computational logic, Memo ATP-25, Automatic Theorem Proving Project, University of Texas, Austin, TX (May 1975).

[Lankford, 1979] D. S. Lankford, On proving term rewriting systems are Noetherian, Memo MTP-3, Mathematics Department, Louisiana Tech. University, Ruston, LA. Revised October 1979 (May 1979).

[Lankford, 1981] D. S. Lankford, A simple explanation of inductionless induction, Memo MTP-14, Department of Mathematics, Louisiana Tech. University, Ruston, LA (August 1981).

[Lankford-etal, 1984] D. Lankford, G. Butler and A. Ballantyne, A progress report on new decision algorithms for finitely presented Abelian groups, R. E. Shostak, ed., Napa, CA. Available as Vol. 170, *Lecture Notes in Computer Science*, Springer, Berlin, pp. 128-141 (1984).

[Lassez-etal, 1988] J.-L. Lassez, M. J. Maher and K. Marriott, Unification revisited, J. Minker, ed., in: *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, Los Altos, CA, pp. 587-625 (1988).

[Lautemann, 1988] C. Lautemann, A note on polynomial interpretation, *Bulletin of the European Association for Theoretical Computer Science* **36**, pp. 129-131 (October 1988).

[LeChenadec, 1985] P. Le Chenadec, Canonical Forms in Finitely Presented Algebras, Pitman-Wiley, London (1985).

[LeChenadec, 1987] P. Le Chenadec, Analysis of Dehn's algorithm by critical pairs, *Theoretical Computer Science* **51** (1,2), pp. 27-52 (1987).

[Lescanne, 1983] P. Lescanne, Computer experiments with the Reve term rewriting system generator, *Proc. of the Tenth ACM Symposium on Principles of Programming Languages*, Austin, TX, pp. 99-108 (January 1983).

[Lescanne, 1984] P. Lescanne, Uniform termination of term-rewriting systems with status, *Proc. of the Ninth Colloquium on Trees in Algebra and Programming*, Cambridge University Press, Bordeaux, France (March 1984).

[Loos, 1981] R. Loos, Term reduction systems and algebraic algorithms, *Proc. of the Fifth GI Workshop on Artificial Intelligence*, Bad Honnef, West Germany, pp. 214-234. Available as Informatik Fachberichte, Vol. 47 (1981).

[Maher, 1988] M. J. Maher, Complete axiomatizations of the algebras of the finite, rational and infinite trees, *Proc. of the Third IEEE Symposium on Logic in Computer Science*, Edinburgh, UK, pp. 348-357 (July 1988).

[Makanin, 1978] J. Makanin, The problem of solvability of equations in a free semi-group, *American Mathematical Society*. Also in Math. USSR Sbornik 32 (2), 1977 (1978).

[Manna-Ness, 1970] Z. Manna and Steven Ness, On the termination of Markov algorithms, *Proc. of the Third Hawaii International Conference on System Science*, Honolulu, HI, pp. 789-792 (January 1970).

[Markov, 1947] A. A. Markov, The impossibility of some algorithms in the theory of associative systems, *Doklady Akademii Nauk SSSR* **55** (7), pp. 587-590. In Russian (1947).

[Martelli-Montanari, 1982] A. MARTELLI and U. MONTANARI, An efficient unification algorithm, *Transactions on Programming Languages and Systems* **4** (2), pp. 258-282 (April 1982).

[Martelli-etal, 1989] A. MARTELLI, G. F. ROSSI and C. MOISO, Lazy unification algorithms for canonical rewrite systems, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 245-274 (1989).

[Martin, 1987] U. MARTIN, How to choose the weights in the Knuth-Bendix ordering, *Proc. of the Second International Conference on Rewriting Techniques and Applications*, P. Lescanne, ed., Bordeaux, France, pp. 42-53. Available as Vol. 256, *Lecture Notes in Computer Science*, Springer, Berlin (May 1987).

[Matijasevic, 1967] J. V. MATIJASEVIC, Simple examples of undecidable associative calculi, *Soviet Mathematics (Dokladi)* **8** (2), pp. 555-557 (1967).

[Matijasevic, 1970] J. V. MATIJASEVIC, ?, *Soviet Mathematics (Dokladi)* (1970).

[McKenzie-etal, 1987] R. MCKENZIE, G. F. MCNULTY and W. TAYLOR, Algebras, Lattices, Varieties, **I**, Wadsworth, Monterey, CA (1987).

[McNulty, 1989] G. F. MCNULTY, An equational logic sampler, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin, pp. 234-262 (April 1989).

[Meseguer-Goguen, 1985] J. MESEGUER and J. A. GOGUEN, Initiality, induction and computability, in: *Alegbraic Methods in Semantics*, M. Nivat, J. Reynolds, ed., Cambridge University Press, pp. 459-540, Cambridge (1985).

[Meseguer-etal, 1989] J. MESEGUER, J. A. GOGUEN and G. SMOLKA, Order-sorted unification, *J. of Symbolic Computation.* To appear (1989).

[Metivier, 1983] Y. METIVIER, About the rewriting systems produced by the Knuth-Bendix completion algorithm, *Information Processing Letters* **16** (1), pp. 31-34 (January 1983).

[Muller-Socher-Ambrosius, 1988] J. MÜLLER and R. SOCHER-AMBROSIUS, Topics in completion theorem proving, SEKI-Report SR-88-13, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, West Germany (1988).

[Musser, 1980] D. R. MUSSER, On proving inductive properties of abstract data types, *Proc. of the Seventh ACM Symposium on Principles of Programming Languages*, Las Vegas, NV, pp. 154-162 (1980).

[Nash-Williams, 1963] C. ST. J. A. NASH-WILLIAMS, On well-quasi-ordering finite trees, *Proc. of the Cambridge Philosophical Society* **59** (4), pp. 833-835 (October 1963).

[Newman, 1942] M. H. A. NEWMAN, On theories with a combinatorial definition of 'equivalence', *Annals of Mathematics* **43** (2), pp. 223-243 (1942).

[Nipkow-Weikum, 1982] T. NIPKOW and G. WEIKUM, A decidability result about sufficient-completeness of axiomatically specified abstract data types, *Proc. of the Sixth GI Conference on Theoretical Computer Science*, Cremers, Kreigel, ed., pp. 257-268. Available as Vol. 145, *Lecture Notes in Computer Science*, Springer, Berlin (1982).

[O'Donnell, 1977a] M. J. O'DONNELL, Computing in systems described by equations, **58**, in: *Lecture Notes in Computer Science*, Springer, Berlin, West Germany (1977).

[O'Donnell, 1977b] M. J. O'DONNELL, Subtree replacement systems: A unifying theory for recursive equations, LISP, Lucid and Combinatory Logic, *Proc. of the Ninth Annual ACM Symposium on Theory of Computing*, pp. 295-305 (1977).

[Oyamaguchi, 1987] M. OYAMAGUCHI, The Church-Rosser property for ground term rewriting systems is decidable, *Theoretical Computer Science* **49** (1) (1987).

[Paterson-Wegman, 1978] M. S. PATERSON and M. N. WEGMAN, Linear unification, *J. of Computer and System Sciences* **16**, Academic Press, pp. 158-167 (1978).

[Pedersen, 1989] J. PEDERSEN, Morphocompletion for one-relation monoids, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin, pp. 574-578 (April 1989).

[Peterson-Stickel, 1981] G. E. PETERSON and M. E. STICKEL, Complete sets of reductions for some equational theories, *J. of the Association for Computing Machinery* **28** (2), pp. 233-264 (April 1981).

[Peterson, 1983] G. E. PETERSON, A technique for establishing completeness results in theorem proving with equality, *SIAM J. on Computing* **12** (1), pp. 82-100 (February 1983).

[Plaisted, 1978] D. A. PLAISTED, Well-founded orderings for proving termination of systems of rewrite rules, Report R-78-932, Department of Computer Science, University of Illinois, Urbana, IL (July 1978).

[Plaisted, 1985] D. A. PLAISTED, Semantic confluence tests and completion methods, *Information and Control* **65** (2/3), pp. 182-215 (May/June 1985).

[Plotkin, 1972] G. PLOTKIN, Building in equational theories, *Machine Intelligence 7*, B. Meltzer, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 73-90 (1972).

[Post, 1947] E. L. POST, Recursive unsolvability of a problem of Thue, *J. of Symbolic Logic* **13**, pp. 1-11 (1947).

[Presburger, 1927] M. PRESBURGER, Über de Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchen, die Addition als einzige Operation hervortritt, *Comptes Rendus du Premier Congrès des Mathématicienes des Pays Slaves*, Warsaw, pp. 92-101, 395 (1927).

[Puel, 1989] L. PUEL, Using unavoidable sets of trees to generalize Kruskal's theorem, *J. of Symbolic Computation.* To appear (1989).

[Puel, 1989] L. PUEL, Embedding with patterns and associated recursive ordering, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, pp. 371-387, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin (April 1989).

[Raoult-Vuillemin, 1980] J.-C. RAOULT and J. VUILLEMIN, Operational and semantic equivalence between recursive programs, *J. of the Association of Computing Machinery* **27** (4), pp. 772-796 (October 1980).

[Raoult, 1981] J.-C. RAOULT, Finiteness results on rewriting systems, *R. A. I. R. O. Theoretical Informatics* **15** (4), pp. 373-391 (1981).

[Raoult, 1984] J.-C. RAOULT, On graph rewritings, *Theoretical Computer Science* **32** (1,2), pp. 1-24 (July 1984).

[Rety-etal, 198?] P. RÉTY, G. SMOLKA and W. NUTT, Narrowing ???, Report,

[Reddy, 1986] U. S. REDDY, On the relationship between logic and functional languages, in: *Logic Programming: Functions, Relations, and Equations*, D. DeGroot, G. Lindstrom, ed., Prentice-Hall, Englewood Cliffs, NJ, pp. 3-36 (1986).

[Remy, 1982] J.-L. RÉMY, Etude des systèmes de réécriture conditionnels et applications aux types abstraits algébriques, Thèse, Institut National Polytechnique de Lorraine, Nancy, France (July 1982).

[Robinson-Wos, 1969] G. ROBINSON and L. WOS, Paramodulation and theorem-proving in first order theories with equality, *Machine Intelligence 4*, B. Meltzer, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 135-150 (1969).

[Robinson, 1965] J. A. ROBINSON, A machine-oriented logic based on the resolution principle, *J. of the Association for Computing Machinery* **12** (1), pp. 23-41 (January 1965).

[Rosen, 1973] B. K. ROSEN, Tree-manipulating systems and Church-Rosser theorems, *J. of the Association for Computing Machinery* **20** (1), pp. 160-187 (January 1973).

[Rusinowitch, 1987] M. RUSINOWITCH, Path of subterms ordering and recursive decomposition ordering revisited, *J. of Symbolic Computation* **3** (1&2) (February/April 1987).

[Rusinowitch, 1989] M. RUSINOWITCH, Démonstration Automatique: Techniques de réécriture, InterEditions, Paris, France (1989).

[Schmidt-Schauss, 1988] M. SCHMIDT-SCHAUSS, Unification in a combination of arbitrary disjoint equational theories, *Proc. of the Ninth International Conference on Automated Deduction*, E. Lusk, R. Overbeek, ed., Argonne, Illinois. Available as Vol. 310, *Lecture Notes in Computer Science*, Springer, Berlin, pp. 378-396 (May 1988).

[Sethi, 1974] R. SETHI, Testing for the Church-Rosser property, *J. of the Association for Computing Machinery* **21**, pp. 671-679. Erratum: Vol. 22, p. 424 (1974).

[Shostak, 1979] R. E. SHOSTAK, An efficient decision procedure for arithmetic with function symbols, *J. of the Association for Computing Machinery* **26** (2), pp. 351-360 (April 1979).

[Siekmann, 1984] J. SIEKMANN, Universal unification, *Proc. of the Seventh International Conference on Automated Deduction*, R. E. Shostak, ed., Napa, CA, pp. 1-42. Available as Vol. 170, *Lecture Notes in Computer Science*, Springer, Berlin (May 1984).

[Sivakumar, 1986] G. SIVAKUMAR, Personal communication (1986).

[Smolka-etal, 1989] G. SMOLKA, W. NUTT, J. A. GOGUEN and J. MESEGUER, Order-sorted equational computation, in: *Resolution of Equations in Algebraic Structures*, H. Ait-Kaci, M. Nivat, ed., **II: Rewriting Techniques**, Academic Press, New York, pp. 299-369 (1989).

[Snyder, 1989] W. SNYDER, Efficient ground completion: An *O(n log n)* algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin, pp. 419-433 (April 1989).

[Stickel, 1976] M. E. STICKEL, The inadequacy of primitive recursive complexity measures for determining finite termination of sets of reductions, Unpublished manuscript, University of Arizona, Tucson, AZ (December 1976).

[Stickel, 1981] M. E. STICKEL, A unification algorithm for associative-commutative functions, *J. of the Association for Computing Machinery* **28** (3), pp. 423-434 (1981).

[Stickel, 1986] M. E. STICKEL, The KLAUS automated deduction system, *Proc. of the Eighth International Conference on Automated Deduction*, Oxford, England, pp. 703-704. Available as Vol. 230, *Lecture Notes in Computer Science*, Springer, Berlin, J. H. Siekmann, ed. (July 1986).

[Stone, 1936] M. STONE, The theory of representations for Boolean algebra, *Transactions of the American Mathematical Society* **40**, pp. 37-111 (1936).

[Szabo, 1982] P. SZABO, Unifikationstheorie erster Ordnung, Thesis, Fakultat fur Informatik, University Karlsruhe, Karlsruhe, West Germany (1982).

[Takeuti, 1987] G. TAKEUTI, Proof Theory, North-Holland. Revised edition (1987).

[Tarski-Givant, 1985] A. TARSKI and S. GIVANT, A Formalization of Set Theory Without Variables, in: *Colloquium Publications* **41**, American Mathematical Society, Providence, RI (1985).

[Tarski, 1951] A. TARSKI, A Decision Method for Elementary Algebra and Geometry, University of California Press, Berkeley, CA (1951).

[Tarski, 1968] A. TARSKI, Equational logic and equational theories of algebras, in: *Contribution to Mathematical Logic*, K. Schutte, ed., North-Holland, Amsterdam (1968).

[Taylor, 1979] W. TAYLOR, Equational logic, in: *Universal Algebra*, G. Grätzer, ed., Springer, New York, pp. 378-400. Second edition (1979).

[Thiel, 1984] J. J. THIEL, Stop losing sleep over incomplete data type specifications, *Proc. of the Eleventh ACM Symposium on Principles of Programming Languages*, Salt Lake City, UT, pp. 76-82 (January 1984).

[Thue, 1914] A. THUE, Probleme über Veranderungen von Zeichenreihen nach gegeben Regeln, *Skr. Vid. Kristianaia I. Mat. Naturv. Klasse* **10/34** (1914).

[Tourlakis, 1984] G. J. TOURLAKIS, Computability, Reston, Reston, VA (1984).

[Toyama, 1987a] Y. TOYAMA, On the Church-Rosser property for the direct sum of term rewriting systems, *J. of the Association for Computing Machinery* **34** (1), pp. 128-143 (January 1987).

[Toyama, 1987b] Y. TOYAMA, Counterexamples to termination for the direct sum for the direct sum of term rewriting systems, *Information Processing Letters* **25**, pp. 141-143 (1987).

[Toyama-etal, 1989] Y. TOYAMA, J. W. KLOP and H. P. BARENDREGT, Termination for the direct sum of left-linear term rewriting systems, *Proc. of the Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, N. Dershowitz, ed., pp. 477-491. Available as Vol. 355 *Lecture Notes in Computer Science*, Springer, Berlin (April 1989).

[Walther, 1988] C. WALTHER, Many-sorted unification, *J. Assoc. Comput. Mach.* **35** (1), pp. 1-17 (January 1988).

[Winkler-Buchberger, 1983] F. WINKLER and B. BUCHBERGER, A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm, *Proc. of the Colloquium on Algebra, Combinatorics and Logic in Computer Science*, Györ, Hungary (September 1983).

[Wirsing, 1989] M. WIRSING, Algebraic specification, in: *Handbook of Theoretical Computer Science*, J. van Leeuwen, ed., North-Holland, Amsterdam (1989).

[Yelick, 1987] K. A. YELICK, Unification in combinations of collapse-free regular theories, *J. of Symbolic Computation* **3** (1&2), pp. 153-181 (February/April 1987).

[Zhang, 1988] H. ZHANG, Reduction, superposition & induction: automated reasoning in an equational logic, Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY (August 1988).

[Zhegalkin, 1927] I. I. ZHEGALKIN, On a technique of evaluation of propositions in symbolic logic, *Matematicheskii Sbornik* **34** (1), pp. 9-27 (1927).

[Anatharaman-Hsiang, 1990] ANATHARAMAN, S. and J. HSIANG, Identities in alternative rings, *J. of Automated Reasoning* **6** (1990) 79–109.

[Bachmair-Ganzinger, 1990] BACHMAIR, L. and H. GANZINGER, On restrictions of ordered paramodulation with simplification, in: *Proc. Second International Workshop on Conditional and Typed Rewriting Systems*, Montreal, Canada, Lecture Notes in Computer Science (Springer, Berlin, to appear).

[Baeten-etal, 1987] BAETEN, J.C.M., J.A. BERGSTRA and J.W. KLOP, Priority rewrite systems, in: *Proc. Second International Conference on Rewriting Techniques and Applications*, Bordeaux, France, Lecture Notes in Computer Science **256**, (Springer, Berlin, 1987) 83–94.

[Bertling-Ganzinger, 1989] BERTLING, H. and H. GANZINGER, Completion-time optimization of rewrite-time goal solving, in: *Proc. Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, Lecture Notes in Computer Science **355** (Springer, Berlin, 1989) 45–58.

[Book, 1982] BOOK, R.V., Confluent and other types of Thue systems, *J. Assoc. Comput. Machin.* **29**(1) (1982) 171–182.

[Borger, 1990] BÖRGER, E., *Computability, Complexity, Logic*, Studies in Logic and the Foundations of Mathematics **128** (North-Holland, Amsterdam, 1990).

[Boudet-etal, 1990] BOUDET, A., E. CONTEJEAN and H. DEVIE, A new AC unification algorithm with a new algorithm for solving Diophantine equations, Tech. Rep., Univ. de Paris-Sud, Orsay, France, 1990.

[Boudet, 1990] BOUDET, A., Unification dans les mélanges de théories equationnelles, Thèse de Doctorat, Univ. de Paris-Sud, Orsay, Paris, 1990.

[Brand-etal, 1979] BRAND, D., J.A. DARRINGER and W.J. JOYNER, JR., Completeness of Conditional Reductions, in: *Proc. Fourth Workshop on Automated Deduction*, Austin, TX (1979).

[Breazu-Tannen, 1990] BREAZU-TANNEN, V. and J. GALLIER, Polymorphic rewriting conserves algebraic strong normalization, *Theoret. Comput. Sci.* (to appear).

[Chew, 1980] CHEW, P., An improved algorithm for computing with equations, in *Proc. Twenty-First IEEE Symposium on Foundations of Computer Science*, Lake Placid, NY (1980) 108–117.

[Comon, 1988] COMON, H., Unification et disunification: Théorie et applications, Thèse de Doctorat, Institut Polytechnique de Grenoble, Grenoble, France, 1988.

[Comon, 1990] COMON, H., Disunification: A survey, in: J. Lassez and G. Plotkin, eds., *Computational Logic: Essays in Honour of Alan Robinson* (MIT Press, Cambridge, MA, to appear).

[Courcelle, 1990] COURCELLE, D., Graph rewriting (to appear).

[Dauchet-etal, 1990] DAUCHET, M., T. HEUILLARD, P. LESCANNE and S. TISON, Decidability of the confluence of finite ground term rewriting systems and of other related term rewriting systems, *Inform. and Comput.* (to appear).

[Dershowitz, 1990] DERSHOWITZ, N., A maximal-literal unit strategy for Horn clauses, in: *Proc. Second International Workshop on Conditional and Typed Rewriting Systems*, Montreal, Canada, Lecture Notes in Computer Science (Springer, Berlin, to appear).

[Dershowitz-Jouannaud, 1990] DERSHOWITZ, N. and J.-P. JOUANNAUD, Notations for rewriting, *Bull. European Assoc. Theoret. Comput. Sci.* (Autumn, 1990).

[Dershowitz-Okada, 1990] DERSHOWITZ, N. and M. OKADA, A rationale for conditional equational rewriting, *Theoret. Comput. Sci.* (to appear).

[Filman, 1978] FILMAN, R.E., Personal communication, 1978.

[Gallier-Snyder, 1990] GALLIER, J. and W. SYNDER, Designing unification procedures using transformations: A survey, in: *Proc. Workshop on Logic from Computer Science*, Berkeley, CA (1990).

[Gallier-Snyder, 1989] GALLIER, J. and W. SYNDER, Complete sets of transformations for general E-Unification, *Theoret. Comput. Sci.* **67** (1989) 203–260.

[Geser, 1989] GESER, A., *Termination Relative*, Doctoral Dissertation, Univ. Passau, Passau, 1989.

[Gnaedig-etal, 1988] GNAEDIG, I., C. KIRCHNER and H. KIRCHNER, Equational completion in order-sorted algebras, in: *Proc. Thirteenth Colloquium on Trees in Algebra and Programming*, Nancy, France (1988) 165–184.

[Halpern-etal, 1990] HALPERN, J.Y., J.H. WILLIAMS and E.L. WIMMERS, Completeness of rewrite rules and rewrite strategies for FP, *J. Assoc. for Comput. Mach.* **37**(1) (1990) 86–143.

[Huet-Levy, 1990] HUET, G. and J.-J. LÉVY, Computations in orthogonal term rewriting systems, in: J. Lassez and G. Plotkin, eds., *Computational Logic: Essays in Honour of Alan Robinson*, (MIT Press, Cambridge, MA, to appear).

[Jaffar, 1990] JAFFAR, J., Minimal and complete word unification, *J. Assoc. for Comput. Mach.* **37**(1) (1990) 47–85.

[Jouannaud-Kirchner, 1990] JOUANNAUD, J. and C. KIRCHNER, Solving equations in abstract algebras: A rule-based survey of unification, in: J. Lassez and G. Plotkin, eds., *Computational Logic: Essays in Honour of Alan Robinson* (MIT Press, Cambridge, MA, to appear).

[Kandri-Rody-etal, 1989] KANDRI-RODY, A., D. KAPUR and F. WINKLER, Knuth-Bendix procedure and Buchberger algorithm—A synthesis, Tech. Rep. 89-18, Dep. of Computer Science, State Univ. of New York, Albany, NY, 1989.

[Kaplan-Remy, 1989] KAPLAN, S. and J.-L. RÉMY, Completion algorithms for conditional rewriting systems, in: H. Ait-Kaci and M. Nivat, eds., *Resolution of Equations in Algebraic Structures* (Academic Press, Boston, 1989) 141–170.

[Kapur-Musser, 1986] KAPUR, D. and D.R. MUSSER, Inductive reasoning for incomplete specifications, in: *Proc. IEEE Symposium on Logic in Computer Science*, Cambridge, MA, (1986) 367–377.

[Kirchner, 1988] KIRCHNER, C., Order-sorted equational unification, Report 954, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, France, 1988.

[Klop, 1980] KLOP, J.-W., *Combinatory Reduction Systems*, Mathematical Centre Tracts **127** (Mathematisch Centrum, Amsterdam, 1980).

[Kounalis, 1985] KOUNALIS, E., Completeness in data type specifications, in: *Proc. EUROCAL Conference*, Linz, Austria (1985).

[Kuchlin, 1985] KÜCHLIN, W., A confluence criterion based on the generalized newman lemma, in: *Proc. EUROCAL Conference*, Linz, Austria (1985).

[Lankford-etal, 1984] LANKFORD, D., G. BUTLER and A. BALLANTYNE, A progress report on new decision algorithms for finitely presented Abelian groups, in: *Proc. Seventh International Conference on Automated Deduction*, Napa, CA, Lecture Notes in Computer Science **170** (Springer, Berlin, 1984) 128–141.

[Martin, 1989] MARTIN, U., A geometrical approach to multiset orderings, *Theoret. Comput. Sci.* **67** (1989) 37–54.

[Middeldorp, 1989] MIDDELDORP, A., Modular aspects of properties of term rewriting systems related to normal forms, in: *Proc. Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, NC, Lecture Notes in Computer Science **355** (Springer, Berlin, 1989) 263–277.

[Okada-Steele, 1990] OKADA, M. and A. STEELE, Ordering structures and the Knuth-Bendix completion algorithm, Unpublished manuscript, Dept. Computer Science, Concordia Univ., Montreal, Canada.

[Padawitz, 1990] PADAWITZ, P., Horn logic and rewriting for functional and logic program design, Report MIP-9002, Fakultät für Mathematik und Informatik, Univ. Passau, Passau, West Germany, 1990.

[Paul, 1985] PAUL, E., Equational methods in first order predicate calculus, *J. of Symbolic Computation* **1**(1) (1985) 7–29.

[Plaisted, 1978] PLAISTED, D.A., A recursively defined ordering for proving termination of term rewriting systems, Report R-78-943, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, 1978.

[Plotkin, 1970] PLOTKIN, G., Lattice theoretic properties of subsumption, Tech. Rep. MIP-R-77, Univ. of Edinburgh, Edinburgh, Scotland, 1970.

[Ruzicka-Privara, 1989] RUZICKA, P. and I. PRIVARA, An almost linear Robinson unification algorithm, *Acta Informat.* **27** (1989) 61–71.

[Sekar-Ramakrishnan, 1990] SEKAR, R.C. and I.V. RAMAKRISHNAN, Programming in equational logic: Beyond strong sequentiality, in: *Proc. Fifth IEEE Symposium on Logic in Computer Science*, Philadelphia, PA (1990).

[Sivakumar, 1986] SIVAKUMAR, G., Personal communication, 1986.

[Slagle, 1974] SLAGLE, J.R., Automated theorem-proving for theories with simplifiers, commutativity, and associativity, *J. Assoc. for Comput. Mach.* **21**(4) (1974) 622–642.

[Zhang-Kapur, 1988] ZHANG, H. and D. KAPUR, First-order theorem proving using conditional equations, in: *Proc. Ninth International Conference on Automated Deduction*, Argonne, Illinois, Lecture Notes in Computer Science **310** (Springer, Berlin, 1988) 1–20.