Computing, Suppl. 4, 11 – 43 (1982)

Algebraic Simplification

B. Buchberger, Linz, and **R. Loos,** Karlsruhe

Abstract

Some basic techniques for the simplification of terms are surveyed. In two introductory sections the problem of canonical algebraic simplification is formally stated and some elementary facts are derived that explain the fundamental role of simplification in computer algebra. In the subsequent sections two major groups of simplification techniques are presented: special techniques for simplifying terms over numerical domains and completion algorithms for simplification with respect to sets of equations. Within the first group canonical simplification algorithms for polynomials, rational expressions, radical expressions and transcendental expressions are treated (Sections 3 – 7). As examples for completion algorithms the Knuth-Bendix algorithm for rewrite rules and an algorithm for completing bases of polynomial ideals are described (Sections 8 – 11).

1. The Problem of Algebraic Simplification

The problem of simplification has two aspects: obtaining *equivalent but simpler* objects and computing *unique representations for equivalent objects.*

More precisely, let T be a class of (linguistic) objects ("*expressions*") as, for instance, (first-order) terms, (restricted classes of) logical formulae, or (restricted classes of) programs and let $\sim$ be an *equivalence relation* on T, for instance functional equivalence, equality derivable from axioms, equality valid in certain models of axioms, congruence modulo an ideal etc.

The *problem of obtaining equivalent but simpler objects* consists in finding an effective procedure S that maps T in T and meets the following specification: For all objects t in T

(SE) $\qquad S(t) \sim t$ and

(SS) $\qquad S(t) \leqslant t.$

Here, $\leqslant$ is the concept of *simplicity* one is interested in (for instance, "$s \leqslant t$" might be "s is shorter than t", "s needs less memory for representation in computer storage than t", "the evaluation of s is less complex than that of t", "numerically, s is more stable than t", or "the structure of s is more intelligible than that of t" etc.). Of course, $S(t) < t$ will be required for "most" t.

The *problem of computing unique representations for equivalent objects* (= the problem of *canonical simplification*) consists in finding an effective procedure S (a "*canonical simplifier*" or "ample function" for $\sim$ (on T)) that maps T in T and

meets the following specifications: For all objects s, t in T

(SE) $S(t) \sim t$ and

(SC) $s \sim t \Rightarrow S(s) = S(t)$,

(i.e. S singles out a unique representative in each equivalence class. $S(t)$ is called a *canonical form* of t).

A well known example of a canonical simplifier is the transformation of elementary arithmetical expressions into "polynomial" form: $x^2 - 1$ is the polynomial form of $(1/2)(2x + 2)(x - 1)$, for instance.

The two types of problems are not totally independent: a consequent iterative reduction of the size of expressions with respect to some measure of "simplicity" may, sometimes, well establish a canonical simplifier or at least give an idea how to construct a canonical simplifier. Conversely, a canonical simplifier trivially defines a corresponding notion of simplicity: the canonical form of an object may be called "simpler" than the object itself. On the other hand, practical procedures that intuitively "simplify" expressions may lead to distinct simplified forms for two equivalent expressions and, conversely, canonical forms of simple expressions may be quite "complex" (for instance, the canonical form of $(x + 2)^5$ is $x^5 + 10x^4 + 40x^3 + 80x^2 + 80x + 32$).

In this paper we shall exclusively be concerned with *canonical simplification* of first-order *terms* (= *algebraic* simplification) and only briefly survey the literature on simplification of other types of expressions and on non-canonical simplification at the end of this section.

For characterizing a canonical simplification problem a careful analysis of the notion of *equivalence* involved is vital. The phrases set in italics in the following paragraphs represent a few examples of frequent equivalence concepts.

As linguistic objects, the elementary arithmetical expressions $(x + 2)^5$ and $x^5 + 10x^4 + 40x^3 + 80x^2 + 80x + 32$ are not *identical*. They are equivalent, however, in the sense that they *represent the same polynomial* in $\mathbb{R}[x]$. $x^6 - 1$ and $x - 1$ do not represent the same polynomial in $\mathbb{R}[x]$ nor in $GF(2)[x]$. They are, however, *functionally equivalent* (i.e. describe the same function) over $GF(2)$. The rational expression $(x^6 - 1)/((x - 3)(x^2 - 1))$ is not functionally equivalent to $(x^4 + x^2 + 1)/(x - 3)$ over $\mathbb{Q}$ (consider the values of the two expressions at $x := \pm 1$). These expressions *denote the same element* in $\mathbb{Q}(x)$, however, and are functionally *equivalent as meromorphic functions*. Similarly, the radical expressions $(x - y)/(\sqrt{x} + \sqrt{y})$ and $\sqrt{x} - \sqrt{y}$ are not functionally equivalent over $\mathbb{Q}$ in a strong sense, but are equivalent as meromorphic functions and may be conceived as denoting the same element in the extension field $\mathbb{Q}(x, y)[\sqrt{x}, \sqrt{y}]$, which is isomorphic to $\mathbb{Q}(x, y)[z_1, z_2]$ modulo the polynomial ideal generated by $z_1^2 - x$ and $z_2^2 - y$. Here *congruence* modulo an ideal appears as yet another notion of equivalence. The *equality* Append(Cons(A, Cons(B, Null)), Cons(C,Null)) = Cons(A, Cons(B, Cons(C, Null))) *may be derived* from the two axioms Append(Null, x) = x, Append(Cons(x, y), z) = Cons(x, Append(y, z)) by mere

"equational reasoning" and is valid in all models of these axioms. In this sense the expressions on the two sides of the equality might be called equivalent. On the other hand, the equality $\text{Append}(x, \text{Append}(y, z)) = \text{Append}(\text{Append}(x, y), z)$ is not derivable from the above axioms by equational calculus alone (some additional induction principle is needed), though the *equality holds* in all computationally "interesting" models (the "initial" models of the axioms). Thus, the two terms $\text{Append}(x, \text{Append}(y, z))$ and $\text{Append}(\text{Append}(x, y), z)$ are equivalent in a sense that is distinct from the preceding one.

For computer algebra, the problem of constructing *canonical simplifiers* is basic, because it is intimately connected with

> the problem of effectively carrying out operations ("*computing*") in algebraic (factor) domains and
>
> the problem of effectively *deciding equivalence.*

In fact, an effective canonical simplifier immediately yields an algorithm for deciding equivalence and for computing in the factor structure defined by the equivalence. Conversely, a decision procedure for equivalence guarantees the existence of an effective (though, in general, not an efficient) canonical simplifier:

Theorem (Canonical simplification and decidability of equivalence). *Let T be a decidable set of objects and $\sim$ an equivalence relation on T. Then: $\sim$ is decidable iff there exists a canonical simplifier S for $\sim$.* ■

Example. In the commutative semigroup defined by the generators a, b, c, f, s and the defining relations $as = c^2s$, $bs = cs$, $s = f$ the rewrite rules $s \to f$, $cf \to bf$, $b^2f \to af$ constitute a canonical simplifier (this can be established by the methods given in Section 11). $a^5bc^3f^2s^3$ and $a^5b^2c^2s^5$ represent the same element of the semigroup because their canonical forms are identical, namely a^7f^5, whereas cs^2 and c^2s represent different elements because their canonical elements bf^2 and af are distinct.

Proof of the Theorem. "$\Leftarrow$": By (SE) and (SC): $s \sim t \Leftrightarrow S(s) = S(t)$.

"$\Rightarrow$": Let g be a computable function that maps $\mathbb{N}$ onto T. Define: $S(s) := g(n)$, where n is the least natural number such that $g(n) \sim s$. S is computable, because g is. It is clear that (SE) and (SC) are satisfied. ■

Theorem (Canonical simplification and computation). *Let T be a decidable set of objects, R a computable binary operation on T, and $\sim$ an equivalence relation on T, which is a congruence relation with respect to R. Assume we have a canonical simplifier S for $\sim$. Define:*

$$\text{Rep}(T) := \{t \in T \mid S(t) = t\} \quad (\textit{set of "canonical representatives", ample set}),$$

$$R'(s, t) := (S(R(s, t)) \quad (\textit{for all } s, t \in \text{Rep}(T)).$$

Then, $(\text{Rep}(T), R')$ is isomorphic to $(T/\sim, R/\sim)$, $\text{Rep}(T)$ is decidable, and R' is computable. (Here, $R/\sim (C_s, C_t) := C_{R(s,t)}$, where C_t is the congruence class of t with respect to $\sim$). ■

This theorem shows that, having a canonical simplifier for an equivalence relation that is a congruence with respect to a computable operation, one can algorithmically master the factor structure. Of course, for concrete examples computationally more efficient ways of realizing R' will be possible. The theorem is proven by realizing that $i(t) := C_t$ $(t \in \mathrm{Rep}(T))$ defines an isomorphism between the two structures.

Example. The rewrite rules $x^3 \to x^2$ and $x^2y \to x^2$ constitute a canonical simplifier S on $T := \mathbb{Q}[x, y]$ for the congruence relation modulo the ideal I generated by the two polynomials $x^3y - x^3$ and $x^3y - x^2$. (Again, this can be established by the methods developed in Section 11.) $\mathrm{Rep}(T)$ = set of all polynomials f in $\mathbb{Q}[x, y]$, such that no monomial in f is a multiple of x^3 or x^2y. Multiplication of residue classes modulo I, for instance, may be isomorphically modeled in $\mathrm{Rep}(T)$ by $R'(s, t) := S(s \cdot t)$. Thus, $R'(xy + 1, xy - 1) = S((xy + 1) \cdot (xy - 1)) = S(x^2y^2 - 1) = x^2 - 1$. ■

It is not always possible to find a canonical simplifier for an equivalence relation on a given set of expressions. In addition to the practical desire of system users to simplify with respect to various intuitive and "local" simplification criteria, this is a theoretical motivation for dealing with non-canonical simplification, see bibliographic remarks below. Actually, it has been proven that some rather simple classes of expressions have an undecidable (functional) equivalence problem and, hence, no canonical simplifier can be found for them (see Section 5). Accordingly, various notions weaker than, but approximating canonical simplification have been introduced in the literature. The two most common ones are *zero-equivalence* simplification and *regular* simplification.

Zero-equivalence simplification may be defined in a context where the given set of expressions contains an element that plays the role of a zero-element:

A computable mapping from T to T is called *zero-equivalence* (or *normal*) *simplifier* for the equivalence relation $\sim$ on T iff for all t in T:

(SE) $\qquad S(t) \sim t$ and

(SZ) $\qquad t \sim 0 \Rightarrow S(t) = S(0)$.

Of course, canonical simplifiers are normal ones. On the other hand, if there exists a computable operation M on T such that for all s, t in T

(ME) $\qquad s \sim t \Leftrightarrow M(s, t) \sim 0$,

then the existence of a normal simplifier implies the existence of a canonical simplifier for $\sim$: Assume M satisfies (ME) (for instance, M might be an operation that represents subtraction) and S is a normal simplifier for $\sim$. Then $\sim$ is decidable, because $s \sim t \Leftrightarrow M(s, t) \sim 0 \Leftrightarrow S(M(s, t)) = S(0)$ (use (ME), (SE) and (SZ)). Now, the existence of a canonical simplifier is guaranteed by the above theorem on canonical simplification and decidability of equivalence.

The notion of *regular simplification* is used in the context of expressions involving transcendental functions (for instance exp, log etc.). Roughly, a regular simplifier guarantees that all non-rational terms in the simplified expression are algebraically independent. (For details see the chapter on computing in transcendental extensions in this volume).

The conceptual framework concerning *canonical simplification*, decidability of equivalence and effective computation in algebraic structures, outlined above is based on various analyses given in Caviness [17], Moses [94], Caviness, Pollack, Rubald [19], Musser [95], Lausch, Nöbauer [77], Loos [79], and Lauer [76].

The simplification of linguistic objects embraces the simplification of *logical formulae* and the "optimization" of *programs*. Both problems are extremely relevant for the computer-aided design and verification of software. Following common usage, we do not treat these topics under the heading of "computer algebra" in this paper, although we believe that a unified treatment of these subjects will be inevitable and promising in the future. King, Floyd [63], Polak [110], Luckham et al. [83], Gerhart et al. [42], Boyer, Moore [8], Suzuki, Jefferson [133], Reynolds [112], Loveland, Shostak [82] are some sources that give an impression of the wide range of simplification techniques for logical formulae in the field of computer-aided program verification and synthesis.

Simplification procedures for algebraic terms that do not explicitly aim at canonicality, but emphasize actual "simplification" of expressions with respect to various, intuitively appealing criteria of simplicity (*non-canonical simplification*) are of great practical importance in computer algebra systems. Since the user of a computer algebra system will be interested in simplification with respect to quite distinct simplification objectives in different contexts, most computer algebra systems provide various simplification procedures for one and the same class of expressions with a possibility for the user to *choose* the most adequate type of simplification *interactively*. Some of the systems include the possibility for the user to *teach the system* new simplification rules.

An overview on existing (non-canonical) simplification facilities, design objectives and simplification criteria applied in computer algebra systems may be obtained by consulting (the introductory sections of) the following papers: Moses [94], Fateman [35], Fitch [40], Brown [11], Hearn [48], Fateman [37] and Pavelle et al. [104] and the manuals of the computer algebra systems, see also the chapter on computer algebra systems in this volume. Fateman [36] and Hearn [47] are illustrative summaries of two of the most advanced simplifiers in existing systems. Yun, Stoutemyer [140] is the most thorough treatment of (non-canonical) simplification.

Substitution (Moses [94]) and *pattern matching* (Fateman [35]) are two basic general purpose techniques for (non-canonical) simplification of expressions. Substitution is used in both directions: substituting expressions for variables in other expressions and recognizing identical subexpressions in a complex expression by pattern matching. An example of applying these techniques is Stoutemyer's simplifier (Stoutemyer [132]) for expressions involving the absolute value function and related functions. Mathematical and software engineering details of general and special purpose (non-canonical) simplifiers are treated in a big number of articles, some typical of them being Fateman [34], Griss [45], Jenks [60], Lafferty [67]. Still, basic techniques are used from the pioneer works of Fenichel [38], Korsvold [66], Tobey [135] and others.

Some special and challenging types of non-canonical "simplifications" are standardization of expressions for *intellegible output* (see, for instance, Martin [87], Foderaro [41]), and simplification in the sense of "*extracting interesting informations*" from expressions (for instance, information about boundedness, continuity, monotonicity, convexity etc.), see Stoutemyer [131].

2. Terms and Algebras: Basic Definitions

A *signature* (or *arity-function*) is a family of non-negative integers (i.e. a function a: $\mathrm{Def}(a) \to \mathbb{N}_0$). The index set of a (i.e. $\mathrm{Def}(a)$) may be called the set of *function symbols* of a, in short $F(a)$. Furthermore, $F(a,n) := \{f \in F(a)/a(f) = n\}$ (set of n-ary function symbols). (We often shall omit the arguments of operators like F if they are clear from the context, compare C, Term etc. in the sequel).

An *a-algebra* is a family A (with index set $F(a)$) of functions ("*operations*") on a set M, such that for all $f \in F(a)$ the function $A(f)$ "denoted by the function symbol f" is a mapping from $M^{a(f)}$ into M. ($A(f)$ is an element in M if $a(f) = 0$). M is called the *carrier* $C(A)$ of A. By abuse of language, sometimes A will be written in contexts where $C(A)$ should be used.

We assume that the reader is familiar with the following notions: *generating set* for an a-algebra, *homomorphic* (*isomorphic*) *mapping* between two a-algebras and *congruence* relation on an a-algebra.

In the class of all a-algebras, for every set X there exists a "*free* a-algebra A_0 with generating set X" with the following property: $X \subseteq C(A_0)$, A_0 is generated by X, and for all a-algebras A and all mappings $m: X \to C(A)$ there exists a unique homomorphism h from A_0 to A which extends m (*universal property* of free algebras). Up to isomorphisms, A_0 is uniquely determined.

Let X be a (denumerable) set disjoint from $F(a)$ (X is called a set of *variables*). The following *term algebra* $\mathrm{Term}(X,a)$ is an example of a free a-algebra with generating set X: The carrier of $\mathrm{Term}(X,a)$ is the minimal set T' such that

$$x \in X \Rightarrow x \in T',$$

$$t_1, \ldots, t_n \in T', \qquad f \in F(a,n) \Rightarrow f t_1 \cdots t_n \in T'$$

($x, t_1, \ldots, t_n, f t_1 \cdots t_n$ denote strings of symbols here!). Furthermore, for every function symbol f in $F(a,n)$ the function $\mathrm{Term}(X,a)(f)$, i.e. the function denoted by f in the term algebra, is defined by

$$\mathrm{Term}(X,a)(f)(t_1, \ldots, t_n) = f t_1 \cdots t_n \qquad (t_1, \ldots, t_n \in C(\mathrm{Term}(X,a))).$$

(For example, $t := \cdot + xy7$ is a *term* from $\mathrm{Term}(X,a)$, where $X := \{x,y\}$, $F(a) = \{\cdot, +, 7\}$, $a(\cdot) = a(+) = 2$, $a(7) = 0$. More readable notations for terms (infix, parentheses etc.) will be used in the subsequent examples). Concrete representations of terms in computers (for instance as threaded lists) are other examples of free a-algebras. Terms in $\mathrm{Term}(X,a)$ that do not contain variables are called *ground terms* over a.

A homomorphic mapping σ from Term into Term with the property that $\sigma(x) = x$ for almost all $x \in X$ is called *substitution*. If A is an a-algebra, an *assignment* in A is a

homomorphic mapping v from Term into A. For $t \in$ Term, $v(t)$ is called the *value* of t at v. Because of the universal property of Term, a substitution (an assignment) is totally determined by fixing its values for all $x \in X$.

Suppose that X is denumerable and is bijectively enumerated by $x : \mathbb{N} \to X$. Let A be an a-algebra, $t \in$ Term, and n be such that all variables occurring in t appear in $\{x_1, \ldots, x_n\}$. The n-ary *function* $\mathrm{Fun}(A, n, t)$ *described by* t on A is defined by $\mathrm{Fun}(A, n, t)(a_1, \ldots, a_n) = v(t)\,(a_1, \ldots, a_n \in A)$, where v is an assignment that satisfies $v(x_1) = a_1, \ldots, v(x_n) = a_n$. t_1 and t_2 are *functionally equivalent* on A iff $\mathrm{Fun}(A, n, t_1) = \mathrm{Fun}(A, n, t_2)$ (where n is such that the variables occurring in t_1 or t_2 appear in $\{x_1, \ldots, x_n\}$).

Let $E \subseteq \mathrm{Term} \times \mathrm{Term}$. (A pair (a, b) in E will be conceived as the left-hand and right-hand side of an "*equation*" in the subsequent context). Two important equivalence relations on Term are associated with E, which are basic for all algebraic theories: $E \models s = t$ (s and t are "*semantically equal* in the theory E") and $E \vdash s = t$ (s and t are "*provably equal* in the theory E").

Definition of $E \models s = t$. If A is an a-algebra, we say that the equation $s = t$ is *valid* in A (or that A is a *model* for $s = t$) iff for all assignments v in A $v(s) = v(t)$. The set of all a-algebras A, in which all equations $a = b$ of E are valid is called the *variety* $\mathrm{Var}(a, E)$ of E. Now,

$$E \models s = t \text{ iff } s = t \text{ is valid in all } a\text{-algebras of } \mathrm{Var}(a, E). \quad \blacksquare$$

Definition of $E \vdash s = t$. $E \vdash s = t$ iff the formula $s = t$ can be derived in finitely many steps in the following calculus ("*equational calculus*"; a, b, c etc. $\in$ Term):

$$\frac{}{a = b} \qquad \text{for all } (a, b) \in E \text{ ("elements of } E \text{ are } \textit{axioms}\text{")},$$

$$\frac{}{a = a}, \quad \frac{a = b}{b = a}, \quad \frac{a = b,\ b = c}{a = c} \qquad (\textit{reflexivity}, \textit{symmetry}, \textit{transitivity} \text{ of } =),$$

$$\frac{a = b}{\sigma(a) = \sigma(b)} \qquad (\text{for all substitutions } \sigma;\ \textit{substitution rule}),$$

$$\frac{a_1 = b_1, \ldots, a_n = b_n}{f a_1 \cdots a_n = f b_1 \cdots b_n} \qquad (\text{for all } f \in F(a, n);\ \textit{replacement} \text{ rule}). \quad \blacksquare$$

The equational calculus is correct and complete for the above defined notion of semantical equality by the following theorem.

Theorem (Birkhoff 1935).

$$E \models s = t \quad \textit{iff} \quad E \vdash s = t. \quad \blacksquare$$

By this theorem, semantical equality can be semi-decided if E is at least a recursively enumerable set. For more details about the above concepts see the monographies on universal algebra (for instance Cohn [21], Lausch, Nöbauer [77]) and the survey on equational theory (Huet, Oppen [58]) which covers also the extensions of the terminology to many-sorted algebras, introduced in Birkhoff, Lipson [7], and

the literature on abstract data types, see for instance Musser [96]. The proof of Birkhoff's theorem is given in Birkhoff [6], compare also O'Donnell [101].

In the sequel, for fixed E, $=_E$ denotes the equivalence relation defined by: $s =_E t$ iff $E \vdash s = t$.

3. Canonical Simplification of Polynomials and Rational Expressions

The General Concept of Polynomial

Let E be a set of equations and A an a-algebra in Var(E). All terms with "constants denoting elements in A", which can be proven equal by E and the definition of the operations in A, are said to "represent the same E-polynomial over A". More precisely, the algebra of *E-polynomials* over A (with variable set X) Pol(X, a, A, E) is the a-algebra that has as its carrier the factor set $C(\text{Term}(X, a'))/\sim$, where a' and (the congruence) $\sim$ is defined as follows: $F(a') := F(a) \cup \text{Name}(A)$, where Name($A$) (a set of *names* for elements in $C(A)$) is a set that is disjoint from $F(a)$ and X and has the same cardinality as $C(A)$. $a'(f) := a(f)$ for $f \in F(a)$ and $a'(f) := 0$ for $f \in \text{Name}(A)$. Furthermore, for $s, t \in \text{Term}(X, a')$

$$s \sim t \quad \text{iff} \quad (E \cup \text{Op}(A)) \vdash s = t,$$

where Op(A) is the set of axioms that completely describe the operations of A, i.e.,

$$\text{Op}(A) := \{(fi_1 \cdots i_n, i) / f \in F(a, n),\ A(f)(c(i_1), \ldots, c(i_n)) = c(i),$$
$$i_1, \ldots, i_n,\ i \in \text{Name}(A),\ n \in \mathbb{N}\}$$

(c is some bijection from Name(A) onto $C(A)$).

Example. $E :=$ set of axioms for commutative rings, $A :=$ ring of integers.

$$\text{Op}(A) = \{1 \cdot 1 = 1,\ 1 \cdot 2 = 2, \ldots, 5 \cdot 7 = 35, \ldots, 1 + 1 = 2, \ldots,$$
$$(-5) + 7 = 2, \ldots\}.$$

$(3x + 1) \cdot (2 - 3x \cdot x) = -9 \cdot x \cdot x \cdot x - 3 \cdot x \cdot x + 6 \cdot x + 2$ is derivable from E and Op(A) in the equational calculus, hence, the two sides of the equation denote the same E-polynomial over A. If $A := GF(2)$, then $\text{Op}(A) = \{0 \cdot 0 = 0,\ 0 \cdot 1 = 0, \ldots, 1 + 1 = 0\}$. The equation $x \cdot x + 1 = x + 1$ is not derivable from E and Op(A) (proof!). $x \cdot x + 1$ and $x + 1$ do not represent the same E-polynomial over A (although $\text{Fun}(A, 1, x \cdot x + 1) = \text{Fun}(A, 1, x + 1)$). For E as above, Pol($\{x_1, \ldots, x_n\}, a, R, E$) is denoted by $R[x_1, \ldots, x_n]$, usually. ■

Canonical Simplifiers for Polynomials

Let $E :=$ set of axioms for *commutative rings* with 1 and R be an arbitrary commutative ring with 1. It is well known (see, for instance, Lausch, Nöbauer [77], p. 24), that the set of all terms of the form $a_n x^n + \cdots + a_0$ ($a_i \cdots$ (names for) elements in R, $a_n \neq 0$) and the term 0 form a system of canonical forms for $R[x]$ (x^n abbreviates $x \cdot x \cdot \cdots \cdot x$, n times). The set of terms of the form $a_{i_1} \cdot x^{i_1} + \cdots + a_{i_m} x^{i_m}$ ($a_{i_j} \neq 0 \cdots$ (names) of elements from R, $i_j \cdots$ "multi-indices" $\in \mathbb{N}_0^n$, $i_1 > \cdots > i_m$ in the lexicographic ordering, for instance) and the

term 0 form a system of canonical forms for $R[x_1, \ldots, x_n]$ (x^i is an abbreviation for the term $x_1^{i_1} \cdot \cdots \cdot x_n^{i_n}$, if i is a multi-index). It is clear, how a canonical simplifier S:Term$(X, a') \to R[x]$, respectively $\to R[x_1, \ldots, x_n]$, may be constructed by iteratively "applying" the axioms of E and the "operation table" Op(A) as "rewrite rules" (supposing that the operations in R are computable). An alternative system of canonical forms for $R[x_1, \ldots, x_n]$, often used in computer algebra systems, is the "recursive" representation: $(3x^2 + 5)y^2 + (x - 3)y + (x^3 - 2x + 1)$, for instance, is a "*recursive*" *canonical form* of a polynomial (see Collins [22]).

In Lausch, Nöbauer [77], pp. 27 canonical forms of polynomials over *groups*, *distributive lattices* with 0 and 1 and *boolean algebras* for $X := \{x_1, \ldots, x_n\}$ are given. Again, the respective canonical simplifiers essentially consist in iteratively "applying" the axioms defining the respective variety.

Canonical Simplifiers for Rational Expressions

Informally, rational expressions over an integral domain R are those formed from the constants and variables by the symbols for the ring operations and a symbol for "division". The equivalence of such expressions cannot be defined along the above pattern, because the special role of the zero element in division cannot be formulated by an equational axiom. Therefore, rational expressions are considered as representing elements in the quotient field formed from $R[x_1, \ldots, x_n]$. Thus, the simplification problem consists in finding a canonical simplifier S for the equivalence relation $\sim$ defined on $R[x_1, \ldots, x_n] \times (R[x_1, \ldots, x_n] - \{0\})$ by:

$$(f_1, g_1) \sim (f_2, g_2) \qquad \text{iff} \qquad f_1 \cdot g_2 = f_2 \cdot g_1.$$

In the case when R is a field, the quotient field defined above is isomorphic to the field obtained by transcendentally extending R by $x_1, \ldots, x_n$. Note that $r \sim r'$ does not imply that r and r' are functionally equivalent, compare Brown [11], p. 28. For example $(x^2 - 1)/(x - 1) \sim x + 1$, but $(x^2 - 1)/(x - 1)$ is undefined for $x = 1$ whereas $x + 1$ yields 2. However, one can define division of functions in the way it is done in the theory of meromorphic functions (where, in the above example, the value of $(x^2 - 1)/(x - 1) = (x - 1)(x + 1)/(x - 1)$ for $x = 1$ is *defined* to be $x + 1 = 2$). We, then, have $(f_1, g_1) \sim (f_2, g_2)$ iff (the function denoted by f_1/g_1) = (the function denoted by f_2/g_2). Rational expressions t that are not quotients of two polynomials, for instance $(x - x/x^2)/(x + 1)$, are simplified by using rational arithmetic (see the chapter on arithmetic in algebraic domains) on the simplified quotients that appear as subterms in t:

$$(x - x/x^2)/(x + 1) \to (x/1 - 1/x)/((x + 1)/1) \to ((x^2 - 1)/x)/((x + 1)/1)$$
$$\to (x - 1)/x.$$

Let us suppose that $R[x_1, \ldots, x_n]$ is a unique factorization domain with 1, for which there exist three computable functions G, /, l with the following properties:

$G(f, g)$ = a greatest common divisor (GCD) of f and g
(i.e. G satisfies: $G(f, g) | f$, $G(f, g) | g$ and
for all h: if $h | f$ and $h | g$, then $h | G(f, g)$),

$f | g \Rightarrow f \cdot (g/f) = g$ (division),

$l(f)$ is a unit and $f \equiv g \Rightarrow l(f) \cdot f = l(g) \cdot g$
(i.e. the function $s(f) := l(f) \cdot f$ is a canonical simplifier for the relation $\equiv$ defined by: $f \equiv g$ iff f and g are associated elements, i.e. $f = u \cdot g$ for some unit u.
A unit is an element for which there exists an inverse).

In this case the above simplification problem can be solved by the following *canonical simplifier*:

$$S((f,g)) := (c \cdot (f/G(f,g)),\ c \cdot (g/G(f,g))), \quad \text{where} \quad c = l(g/G(f,g)).$$

This solution of the simplification problem for rational expressions is essentially based on procedures for a GCD (see the chapter on polynomial remainder sequences). Since these procedures are relatively time consuming it is important to carefully avoid superfluous calls of GCD in the arithmetic based on the above canonical forms (see the chapter on arithmetic in algebraic domains, algorithms of Henrici [49]). Also, it may be recommendable to store the numerator and denominator of rational expressions in factored or partially factored form, if such factors are known in special contexts (see Brown [11], Hall [46]).

4. Canonical Simplification of Radical Expressions

Roughly, *radical expressions* are terms built from variables $x_1, \ldots, x_n$, constants (for elements in $\mathbb{Q}$), the arithmetical function symbols $+, -, \cdot, /$ and the radical sign $\sqrt[q]{\ }$ or, equivalently, rational powers ("*radicals*") s^r $(r \in \mathbb{Q})$ (s in s^r is called a *radicand*). A natural (modified) notion of functional equivalence for these expressions is as follows:

$$s \sim t \ (s \text{ is } \textit{meromorphically equivalent} \text{ to } t) \quad \text{iff} \quad \mathrm{Fun}'(s) = \mathrm{Fun}'(t).$$

Here, Fun′ essentially is defined as Fun, but $\mathrm{Fun}'(s/t) = \mathrm{Fun}'(s)/'\mathrm{Fun}'(t)$ where $/'$ denotes division of meromorphic functions (see the preceding section) and, furthermore, $\mathrm{Fun}'(s^r)$ (where r denotes a positive rational) is a meromorphic function that satisfies

$$\text{(R)} \qquad \mathrm{Fun}'(s^r)^v - \mathrm{Fun}'(s)^u = 0$$

(where u, v are relatively prime integers with $r = u/v$). Let T be a set of radical terms built from the radicals $s_1^{r_1}, \ldots, s_k^{r_k}$. Suppose that for all $1 \leqslant i \leqslant k$ the equation (R) for $s_i^{r_i}$ is irreducible over $\mathbb{Q}(\mathrm{Fun}'(x_1), \ldots, \mathrm{Fun}'(x_n), \mathrm{Fun}'(s_1^{r_1}), \ldots, \mathrm{Fun}'(s_{i-1}^{r_{i-1}}))$. Then all possible *interpretations* Fun′ of the expressions in T satisfying the above conditions are "differentially isomorphic", i.e. the structure of the field of functions described by expressions in T does not depend on the chosen "branches" of the meromorphic functions defined by (R) ("*regular algebraic field description*", see Risch [116], Caviness, Fateman [18], Davenport [25]).

Thus, $\sim$ on T may be decided by the following procedure:

1. Construct an algebraic extension field $K := \mathbb{Q}(x_1, \ldots, x_n)[\alpha_1, \ldots, \alpha_m]$ such that all terms in T may be conceived as pure arithmetical terms in K, i.e. terms that are composed from constants for elements in K and the arithmetical function symbols $+, -, \cdot, /$.

2. For deciding $s \sim t$ evaluate s and t by applying rational simplification and an effective arithmetic in K. $s \sim t$ iff both evaluations yield the identical result. The construction of minimal irreducible polynomials for $\alpha_1, \ldots, \alpha_m$ is a means for having an effective arithmetic available in K.

An effective evaluator for ground terms over K, hence, is a *canonical simplifier* for $\sim$ on T. (Note, however, that this simplifier depends on the field K constructed in step 1 of the procedure).

For unambiguously specifying the element of K denoted by a given radical term t it is still necessary to state the *conventions* one wants to use for relating subterms of the form $(s \cdot t)^r$ with s^r and t^r. Note that the rule $(s \cdot t)^r = s^r \cdot t^r$ is a convention that attributes a special interpretation to $(s \cdot t)^r$. Other conventions are possible that imply other interpretations. For instance, $\sqrt{(x \cdot x)}$ may equally well be meant to denote $\sqrt{x} \cdot \sqrt{x} = x$ or $\sqrt{(-x)} \cdot \sqrt{(-x)} = -x$ (this corresponds to computing in $\mathbb{Q}(x)[y]$ modulo the two irreducible factors $y - x$ and $y + x$ of $y^2 - x^2$ respectively). Conventions of the kind $\sqrt{(x \cdot x)} = \mathrm{abs}(x)$, though desirable in certain contexts, would lead outside the range of interpretations, for which an isomorphism theorem holds.

A simplification algorithm of the above type has been developed in Caviness [17], Fateman [35], Caviness, Fateman [18] for the special case of unnested radical expressions, i.e. radical expressions whose radicals do not contain other radicals as subterms:

Algorithm (Canonical Simplification of Radical Expressions, Caviness, Fateman 1976).

Input:

$t_1, \ldots, t_l$ (unnested radical expressions in the variables $x_1, \ldots, x_n$).

Output:

1. $M_1, \ldots, M_m$ (minimal irreducible polynomials for algebraic entities $\alpha_1, \ldots, \alpha_m$, such that $t_1, \ldots, t_l$ may be conceived as pure arithmetical ground terms with constants from $K := \mathbb{Q}(x_1, \ldots, x_n)[\alpha_1, \ldots, \alpha_n]$).
2. $s_1, \ldots, s_l$ (s_i is the result of evaluating t_i in K. s_i is taken as the *canonically simplified* form of t_i).

1. Construction of K:

1.1. All $s_i := t_i$.

1.2. Transform all s_i
by rationally *simplifying all radicands* and applying the rule $(p/q)^{-r} \to (q/p)^r$.

1.3. $R :=$ set of all "*radicals*"
in the s_i, i.e. of subterms of the form $(p/q)^r$, where p and q are relatively prime elements in $\mathbb{Z}[x_1, \ldots, x_n]$, q is positive (i.e. the leading coefficient of q is positive), r is a positive rational number $\neq 1$.

1.4. $P :=$ set of all "*polynomial radicands*"
in the s_i, i.e. polynomials p, q for which $(p/q)^r$ is in R.

1.5. $B :=$ a "*basis*" for P,
i.e. a set of irreducible positive polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ of degree $\geqslant 0$ such that
if b is in B then $b | p$ for a p in P and
if p is in P then $p = \pm \prod b_i^{e_i}$ for certain b_i in B and e_i in $\mathbb{N}$.
(A basis B for P can be obtained by factorizing all p in P and collecting all irreducible factors).

1.6. If P contains a non-positive element: Add the element -1 to B.
$m :=$ *number of elements* in B.

1.7. Transform all s_i by applying the following *conventions*:
$(p/q)^r \to p^r/q^r$,
$(\prod b_i^{e_i})^r \to \prod (b_i^{e_i \cdot r})$, $(-\prod b_i^{e_i})^r \to (-1)^r \cdot \prod (b_i^{e_i \cdot r})$,
$b^r \to b^w \cdot b^{u/v}$, if $r = w + u/v$ ($w \in \mathbb{N}_0$, $0 \leqslant u/v < 1$, u, v relatively prime).

1.8. For all b in B determine the *radical degree* $d(b)$:
$d(b) := \operatorname{lcm}(v_1, \ldots, v_k)$, where $u_1/v_1, \ldots, u_k/v_k$ are the reduced rational powers to which b appears in the s_i.

1.9. Transform all s_i by
expressing all $b_j^{u/v} (b_j \in B)$ as *powers of* $b_j^{1/d(b_j)}$ and
replacing all $b_j^{1/d(b_j)}$ by new variables y_j.

1.10. Determine *minimal irreducible polynomials*:
$M_j :=$ an irreducible factor of $y_j^{d(b_j)} - b_j$ over
$\mathbb{Q}(x_1, \ldots, x_n)[\alpha_1, \ldots, \alpha_{j-1}]$, where $\alpha_j :=$ "$b_j^{1/d(b_j)}$".
(If -1 is in B, say $b_1 = -1$, then $\alpha_1 := \omega_{2d(b_1)}$).

2. Determine the canonical forms s_i:

2.1. Iterate:
Rationally simplify all s_i and
execute the arithmetical operations in K, i.e. in
$\mathbb{Q}(x_1, \ldots, x_n)[y_1, \ldots, y_m]$ modulo the polynomials M_j. ■

Example. $t = (((2x-2)/(x^3-1))^{-7/3} + (2/(x+1))^{1/2})/(24x+24)^{1/4}$.
1.2.: $s = (((x^2+x+1)/2)^{7/3} + (2/(x+1))^{1/2})/(24x+24)^{1/4}$.
1.3., 1.4.: $R = \cdots$, $P = \{x^2+x+1, 2, x+1, 24x+24\}$.
1.5.: $B = \{2, 3, x+1, x^2+x+1\}$.
1.7.: $s = ((((x^2+x+1)^2(x^2+x+1)^{1/3})/(2^2 2^{1/3})$
$+ (2^{1/2}/(x+1)^{1/2}))/(2^{3/4} 3^{1/4}(x+1)^{1/4})$.
1.8.: $d(2) = 12$, $d(3) = 4$, $d(x+1) = 4$, $d(x^2+x+1) = 3$.
1.9.: $s = (((x^2+x+1)^2 \cdot y_4)/(4 \cdot y_1^4) + y_1^6/y_3^2)/(y_1^9 \cdot y_2 \cdot y_3)$.
1.10.: $M_1 = y_1^{12} - 2$, $M_2 = y_2^4 - 3$, $M_3 = y_3^4 - (x+1)$, $M_4 = y_4^3 - (x^2+x+1)$.
2.1.: $s = ((x^2+x+1)^2 \cdot y_3^2 \cdot y_4 + 4 \cdot y_1^{10})/(4 \cdot y_1^{13} \cdot y_2 \cdot y_3^3)$
$= ((x^4+2x^3+3x^2+2x+1)/(48x+48)) \cdot 2^{11/12} \cdot 3^{3/4}(x+1)^{3/4}$
$\cdot (x^2+x+1)^{1/3} + (1/(6x+6)) \cdot 2^{9/12} \cdot 3^{3/4} \cdot (x+1)^{1/4}$. ■

Note that the above construction, in general, does not lead to the minimal extension field K of $\mathbb{Q}(x_1, \ldots, x_n)$ in which the above simplification procedure may be carried out.

Several computational improvements of the algorithm and a complexity analysis based on work of Epstein [32] can be found in Caviness, Fateman [18].

(Theoretically, the algorithm is exponential in the number of variables. However, it is feasible for expressions of moderate size.)

An algorithm similar to the Caviness-Fateman algorithm has also been developed in Fitch [39, 40]. The simplification of radicals is also treated in Zippel [142]. In Shtokhamer [127] ideas for "local" simplification of unnested radical expressions are given (i.e. simplification in the style of step 2 prior to constructing the field K for the whole expression). For treating nested radicals effective computation in residue class rings modulo arbitrary polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ is an adequate framework. Proposals in this direction have been made in Kleiman [64], Buchberger [12], Shtokhamer [125, 126], see also Section 11.

5. Canonical Simplification of Transcendental Expressions: Unsolvability Results

Further extension of the expressive power of the term classes considered may lead to term classes whose simplification problem is algorithmically unsolvable. An example is the class $R2$ of terms generated from one variable x, constants for the rationals, π, and the function symbols $+$, $\cdot$, sin, abs, whose simplification problem with respect to functional equivalence $\sim_{\mathbb{R}}$ on $\mathbb{R}$ is algorithmically unsolvable. (It should be clear how the definition of $R2$ and $\sim_{\mathbb{R}}$ can be made precise following the pattern given in Section 2. Expressions involving function symbols for transcendental functions like exp, log, sin, erf are called *transcendental expressions*. The class of rational expressions over $\mathbb{C}$ extended by exp and log is called the class of *elementary transcendental* expressions).

Theorem (Caviness, Richardson, Matijasevic 1968, 1970). *The predicate "$t \sim_{\mathbb{R}} 0$" is undecidable (for $t \in R2$). (Hence, $\sim_{\mathbb{R}}$ is undecidable and, by the theorem on canonical simplification and equivalence, there cannot exist a zero-equivalence or canonical simplifier for $\sim_{\mathbb{R}}$ on $R2$).* ■

Proof (Sketch). The proof proceeds by a number of problem reductions via recursive translators ("m-reductions" in the sense of algorithm theory, Rogers [119]. The following variant of the famous result of Matiyasevic [89] concerning the undecidability of Hilbert's tenth problem serves as a basis of the proof. For a certain n the predicate

(1) "There exist $z_1, \ldots, z_n \in \mathbb{Z}$ such that $\mathrm{Fun}_{\mathbb{Z}}(t)(z_1, \ldots, z_n) = 0$"

is undecidable for $t \in P := \mathbb{Z}[x_1, \ldots, x_n]$. (Actually $n = 13$ is the minimal such n known so far, see Manin [84].)

First problem reduction. There exists a recursive "translator" $T1 : P \to R3$ such that for all $t \in P$:

(2) there exist $z_1, \ldots, z_n \in \mathbb{Z}$ such that $\mathrm{Fun}_{\mathbb{Z}}(t)(z_1, \ldots, z_n) = 0 \Leftrightarrow$
there exist $b_1, \ldots, b_n \in \mathbb{R}$ such that $\mathrm{Fun}_{\mathbb{R}}(T1(t))(b_1, \ldots, b_n) < 0$.

Here, $R3$ essentially is defined as $R2$ with the difference that we allow n variables $x_1, \ldots, x_n$ and that abs does not appear in the terms of $R3$. Using sin and π, a suitable $T1$ may be defined as follows ($t \in P$):

(3) $T1(t) := (n+1)^2 \cdot [t^2 + \sin^2(\pi x_1) \cdot D(1, t)^2 + \cdots + \sin^2(\pi x_n) \cdot D(n, t)^2] - 1$.

Here $D(i,t)$ is a "dominating" function for $\partial/\partial x_i(t^2)$. It is clear that, given $t \in P$, a term $t' \in P$ may be found effectively, such that t' describes $\partial/\partial x_i(t^2)$. Thus, for constructing $D(i,t)$ it suffices to have a recursive function $T: P \to P$ such that, for all $t \in P$, $T(t)$ describes a "dominating" function for t, i.e. a function that satisfies for all $b_1, \ldots, b_n \in \mathbb{R}$, and for all $\Delta_1, \ldots, \Delta_n \in \mathbb{R}$ with $|\Delta_i| < 1$:

(4) $\mathrm{Fun}_{\mathbb{R}}(T(t))(b_1, \ldots, b_n) > 1$ and
$\mathrm{Fun}_{\mathbb{R}}(T(t))(b_1, \ldots, b_n) > |\mathrm{Fun}_{\mathbb{R}}(t)(b_1 + \Delta_1, \ldots, b_n + \Delta_n)|$.

A suitable T may be defined by induction on the structure of the terms in P. Knowing (4), the proof of (2) is easy.

Second problem reduction. There exists a recursive "translator" $T_2: R3 \to R3$ such that for all $t \in R3$:

(5) there exist $b_1, \ldots, b_n \in \mathbb{R}$ such that $\mathrm{Fun}_{\mathbb{R}}(t)(b_1, \ldots, b_n) < 0 \quad \Leftrightarrow$
there exists $b \in \mathbb{R}$ such that $\mathrm{Fun}_{\mathbb{R}}(T2(t))(b) < 0$.

The construction of $T2$ is tricky: it simulates the application of "pairing functions", which normally are used in algorithm theory for the reduction of n-dimensional problems to one-dimensional ones. Since pairing functions are not available in $R3$, the sinus-function is used to partially approximate the effect of pairing functions.

Third problem reduction. There exists a recursive "translator" $T3: R3 \to R2$ such that for all $t \in R3$:

(6) there exists $b \in R$ such that $\mathrm{Fun}_{\mathbb{R}}(t)(b) < 0 \quad \Leftrightarrow$
$\Leftrightarrow$ not $(T3(t) \sim_{\mathbb{R}} 0)$.

A suitable $T3$ is:

(7) $T3(t) := \mathrm{abs}(t) - t$.

Given the unsolvability of the problem stated in (1) and the problem reductions in (2), (5), and (6), it is thus shown that $\sim_{\mathbb{R}}$ is undecidable. ■

The proof method for the above unsolvability result has been developed in Richardson [113]. It has been applied to Matiyasevic's result in Caviness [17]. Matiyasevic's theorem is the culmination of earlier work by M. Davis, J. Robinson and H. Putnam, for instance Davis, Putnam, Robinson [27]. A clear and detailed presentation of Matiyasevic's result is given in Davis [26]. Other unsolvability results concerning transcendental terms in computer algebra may be found in Fenichel [38], Risch [116], Moses [93].

6. (Canonical) Simplification of Transcendental Expressions: Miscellaneous Techniques

In this section we review miscellaneous techniques for canonical simplification of transcendental expressions that have been developed before the technique based on the "structure theorems" (see next section) has been matured into its present state. There are three groups of methods falling into this category: *specific techniques* for relatively limited classes of transcendental expressions, *point-evaluation* (a probabilistic method), and general *reduction methods* (reducing the zero-equivalence problem for a given expression to that of simpler expressions).

Examples for methods in the first group are a zero-equivalence simplifier for the "*rational exponential expressions*" (rational numbers, i, π, $x_1, \ldots, x_n$, $+$, $-$, $\cdot$, $/$, exp) given in Brown [9] and a canonical simplifier for the "*exponential polynomials*" (rational numbers, x, $+$, $-$, $\cdot$, exp) given in Caviness [17], see also Moses [94]. These algorithms, essentially, consist in a systematic application of known identities for exp (as, for instance, $\exp(x \cdot y) = \exp(x) + \exp(y)$) and applying tests for (linear or algebraic) dependencies between subterms. The proof of the canonicality (normality) of these algorithms is based on number theoretic *conjectures* that are plausible although their proofs seem to be extremely difficult. (For instance, Brown uses the conjecture that if $\{q_1, \ldots, q_k, i\pi\}$ is linearly independent over the rational numbers, $\{e^{q_1}, \ldots, e^{q_k}, z, \pi\}$ is algebraically independent over the rational numbers.)

Point-evaluation (Martin [87], Oldehoeft [102], Fitch [40]), may be used for testing functional equivalence of (transcendental) expressions. Theoretically, the idea is simple: If s is not functionally equivalent to t (s, t transcendental expressions), then $\mathrm{Fun}(s - t)$ has, at most, a countable number of solutions, while the real (or complex) numbers are uncountable. Thus, for a point z chosen at random, $\mathrm{Fun}(s - t)(z) = 0$ implies $s \sim t$ "with probability 1". Practically, because of rounding errors, overflow and underflow, floating-point arithmetic is not satisfactory for performing the evaluations necessary in this context. Finite field computations (Martin [87]), interval arithmetic (Fitch [40]) and computing with transcendental ground terms are the alternatives that have been proposed. The latter method, though seemingly natural, cannot be applied in general because very little is known about the algebraic dependence of transcendental ground terms (the *constant problem*) (see the survey by (Lang [68])).

As an example of a general reduction method we describe *Johnson's method* [61]. (A second reduction method, based on the notion of Budan-sequences, has been presented by Richardson [114]). Johnson's method is applicable to any class of transcendental expressions for which a general notion of "*eigenelements*" can be defined. The method presupposes that the zero-equivalence problem can be solved effectively for certain subclasses of the term class considered. Since this is not generally the case in practical examples, Johnson's method is an algorithm in a relativized sense only.

Let R and K ($K \subseteq R$) be classes of transcendental expressions such that $R/\sim$ is a ring without zero divisors and $K/\sim$ is a field, where $\sim$ denotes functional equivalence on R. Let $\phi: R \to R$ and $E: R \to K$ be computable functions such that E determines "*eigenvalues*" $E(e)$ for certain "*eigenelements*" e of R:

$e \in R$ is an *eigenelement* (w.r.t. ϕ) iff

$$\mathrm{not}(e \sim 0) \text{ and } \phi(e) \sim E(e) \cdot e.$$

We assume:

(1) $\phi(s + t) \sim \phi(s) + \phi(t)$.
(2) $\phi(K) \subseteq K$, $\phi(1) \sim 0$.
(3) If e_1, e_2 are eigenelements then e_1 is invertible and e_1^{-1}, $e_1 \cdot e_2$ are eigenelements.

(4) "$t \sim 0$" is decidable for $t \in K$.
(5) "$t \sim 0$" is decidable for $t \in \ker(\phi) := \{t/\phi(t) \sim 0\}$.

From (1)–(3) one can easily prove that

(6) $\phi(0) \sim 0$,
(7) all elements in K are eigenelements,
(8) e is an eigenelement and not $(\phi(e) \sim 0) \Rightarrow \phi(e)$ is an eigenelement.

Algorithm (Test for zero-equivalence. Johnson 1971).

Input: eigenvalues $e_1, \ldots, e_n$ of an effective operator ϕ on R.

Question: Is $S := e_1 + \cdots + e_n \sim 0$?

1. If $n = 1$: Answer "Not($S \sim 0$)". Return.
2. $T := b_1 \cdot e_n^{-1} \cdot e_1 + \cdots + b_{n-1} \cdot e_n^{-1} \cdot e_{n-1}$,
   where $b_i := E(e_n^{-1} \cdot e_i)$ $(i := 1, \ldots, n-1)$.
   [verify $T \sim \phi(e_n^{-1} \cdot S)$; use (1), (2)].
3. If not $(T \sim 0)$: Answer "Not($S \sim 0$)". Return.
   [Use the fact that $\phi(0) \sim 0$.
   Note that "$T \sim 0$" may decided effectively:
   If all $b_i \sim 0$: Answer "$T \sim 0$".
   [Use (4); note that all $b_i \in K$.]
   Otherwise apply the algorithm recursively to T.
   [Note that all $b_i \cdot e_n^{-1} \cdot e_i$ with not $(b_i \sim 0)$ are eigenelements by (3), (7)]].
4. [Case: $T \sim 0$, in this case: $e_n^{-1} \cdot S \in \ker(\phi)$.]
   4.1. If not$(e_n^{-1} \cdot S) \sim 0$: Answer "Not($S \sim 0$)". Return.
   [Note that "$e_n^{-1} \cdot S \sim 0$" can be decided by (5)].
   4.2. Answer "$S \sim 0$". Return. ■

Examples for the application of this algorithm are given in Johnson [61]. For instance, ϕ may be the operation of formal differentiation on a set R of transcendental expressions. In this context, K can be chosen to be the set of rational expressions. The computation of eigenvalues may be effected by using the following rules: $E(r) = 0$, $E(u \cdot v) = E(u) + E(v)$, $E(u/v) = E(u) - E(v)$, $E(u^r) = rE(u)$, $E(s) = s'/s$, $E(e^s) = s'$ ($r \ldots$ rational number; $u, v \ldots$ arbitrary expressions in R; $s \ldots$ rational expression).

7. Canonical Simplification of Transcendental Expressions: Structure Theorems

The most advanced and systematic method of simplifying transcendental expressions t is based on the so-called structure theorems. The basic idea is similar to the procedure followed in Section 4: in a first step a (transcendental) extension field K of $\mathbb{Q}$ is constructed such that t may be conceived as denoting an element in K. Then t is simplified using rational arithmetic. The structure theorems guide the extension procedure. This procedure is described in the chapter on computing in transcendental extensions.

8. Complete Reduction Systems, Critical Pairs, and Completion Algorithms

In classes T of linguistic objects with an equivalence relation $\sim$ one often can define in a natural way a "reduction relation" $\rightarrow$, which describes how "one can get, in one step, from an object s to a more simple but equivalent object t". The *iteration of this reduction* yields a *canonical simplifier* for $\sim$, if $\rightarrow$ can be handled algorithmically, the reflexive, symmetric and transitive closure of $\rightarrow$ is $\sim$, and $\rightarrow$ is *complete* in the sense that

> the iteration of the reduction process always terminates after finitely many steps at an irreducible object (*finite termination* property) and
>
> different reductions processes starting from the same object always terminate at the same irreducible object (*uniqueness*).

For concrete reduction relations $\rightarrow$, the proof of the finite termination property mostly needs particular techniques. The test for uniqueness can often be carried out in an algorithmic way by using two important ideas: *localization* (generally applicable) and the method of *critical pairs* (often applicable for reduction relations that are "generated" from finitely many reduction patterns). Finitely generated reduction relations that are not complete may be completed by adding reduction patterns that arise in the analysis of the critical pairs. This method of *completion* is the third basic idea, which together with localization and critical pair analysis constitutes a powerful and widely applicable methodology for constructing canonical simplifiers.

More formally, let $T \neq 0$ be an arbitrary set (of linguistic objects). In the context of this section an arbitrary binary relation $\rightarrow \subseteq T \times T$ will be called a *reduction* relation. The *inverse* relation, the *transitive closure*, the *reflexive-transitive* closure and the *reflexive-symmetric-transitive* closure of $\rightarrow$ will be denoted by $\leftarrow$, $\rightarrow^+$, $\rightarrow^*$, and $\leftrightarrow^*$, respectively. Also, $\rightarrow^0 =$ identity, $\rightarrow^{n+1} = \rightarrow \circ \rightarrow^n$. If $\rightarrow$ is clear from the context, by $\underline{x}$ we denote that $x \in T$ is in *normal form* with respect to $\rightarrow$ (i.e. there is no $y \in T$ such that $x \rightarrow y$). x and y have a *common successor* (in symbols, $x \downarrow y$) iff for some $z: x \rightarrow^* z \leftarrow^* y$. $\rightarrow$ is called *noetherian* ($\rightarrow$ has the finite termination property) iff there is no infinite chain of the form $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots$.

Now, let $\sim$ be an equivalence relation on T and $\rightarrow$ be a noetherian reduction relation on T such that $\leftrightarrow^* = \sim$. Suppose we have a computable "*selector*" function Sel: $T \rightarrow T$ such that $x \rightarrow \mathrm{Sel}(x)$ for all $x \in T$ that are not in normal form. Consider the computable function S recursively defined by

$$S(x) := \textit{if } x \text{ is in normal form } \textit{then } x \textit{ else } S(\mathrm{Sel}(x)).$$

Let us call an S of this kind a *normal-form algorithm* for $\rightarrow$. Our objective is to provide an algorithmic test for deciding whether S is a canonical simplifier. First, observe that S satisfies (SE), i.e. $S(x) \leftrightarrow^* x$. We even have

(SE′) $$\text{for all } x \in T\text{: } x \rightarrow^* \underline{S(x)}.$$

In the sequel we shall present a number of lemmas showing that the test for (SC), i.e. the property: $x \leftrightarrow^* y \Rightarrow S(x) = S(y)$, can be reduced successively to intuitively

easier tests and finally, can be "*localized*". The method of *critical pairs*, then, will make the test effective. Let S be a normal-form algorithm for $\rightarrow$ ($\rightarrow$ noetherian!).

Lemma (Reduction of canonicality to the Church-Rosser property). *S is a canonical simplifier for $\leftrightarrow^*$ iff $\rightarrow$ has the Church-Rosser property.* ($\rightarrow$ *has the Church-Rosser property iff for all* $x, y \in T$: $x \leftrightarrow^* y \Rightarrow x \downarrow y$). ■

Proof. "$\Rightarrow$": Easy. "$\Leftarrow$": If $x \leftrightarrow^* y$, then for some z we have: $\underline{S(x)} \leftarrow^* x \rightarrow^* \underline{z} \leftarrow^* y \rightarrow^* \underline{S(y)}$ (apply the Church-Rosser property). Again by the Church-Rosser property (applied to $\underline{S(x)}$ and $\underline{z}$), we have $S(x) = z$ and, analogously, $z = S(y)$. ■

Lemma (Reduction of the Church-Rosser property to confluence). $\rightarrow$ *has the Church-Rosser property iff* $\rightarrow$ *is confluent.* ($\rightarrow$ *is confluent iff for all* $x, y, z \in T$: $x \leftarrow^* z \rightarrow^* y \Rightarrow x \downarrow y$). ■

Proof. "$\Rightarrow$": Easy. "$\Leftarrow$": Since $\leftrightarrow^*$ is the union of all $\leftrightarrow^n$ we can use induction on n. $n = 0$: easy. If $x \leftrightarrow^{n+1} y$, then $x \rightarrow z \leftrightarrow^n y$ or $x \leftarrow z \leftrightarrow^n y$ for some z. In each case, by induction hypothesis: $z \rightarrow^* u \leftarrow^* y$ for some u. In the first case, $x \rightarrow^* u \leftarrow^* y$, hence, $x \downarrow y$. In the second case, $x \rightarrow^* v \leftarrow^* u$ for some v by confluence, hence, $x \rightarrow^* v \leftarrow^* u \leftarrow^* y$, i.e. $x \downarrow y$. ■

Lemma (Reduction of confluence to local confluence. Newman 1942). $\rightarrow$ *confluent iff* $\rightarrow$ *locally confluent.* ($\rightarrow$ *is locally confluent iff for all* $x, z, y \in T$: $x \leftarrow z \rightarrow y \Rightarrow x \downarrow y$.) ■

Proof: "$\Rightarrow$": Easy. "$\Leftarrow$": By "noetherian" induction. Let $z_0 \in T$ be arbitrary but fixed. Induction hypothesis: For all z with $z_0 \rightarrow^+ z$: for all x, y: $(x \leftarrow^* z \rightarrow^* y \Rightarrow x \downarrow y)$. We show: For all x, y: $(x \leftarrow^* z_0 \rightarrow^* y \Rightarrow x \downarrow y)$. Case $x = z_0$ and case $z_0 = y$: easy. In the other case: $x \leftarrow^* x_1 \leftarrow z_0 \rightarrow y_1 \rightarrow^* y$ for some x_1, y_1. Then, by local confluence and induction hypothesis, there exist u, v, w with the properties shown in Fig. 1. ■

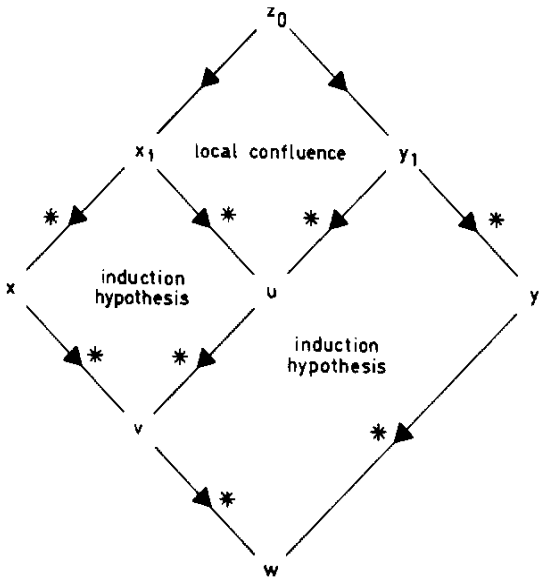

Fig. 1

The above lemmata can also be combined in the following way in order to yield a test for canonicality of S that involves S only:

Lemma (Local test for canonicality). *S is a canonical simplifier for $\leftrightarrow^*$ iff for all x, z, y: $(x \leftarrow z \rightarrow y \Rightarrow S(x) = S(y))$.* ■

Still, this test for canonicality is not effective, because infinitely many x, y, z must be considered, in general. In the next section we will see how this test can be made effective in situations where $\rightarrow$ is generated from finitely many "reduction patterns" by operations as "substitution" and "replacement" (in a very general sense).

Historically, the central role of the Church-Rosser property for canonical simplification has been observed first in the development of λ-calculus, see Church, Rosser [20], Hindley [51], Hindley, Lercher, Seldin [52]. Newman's lemma as a general technique appeared first in Newman [100]. A complete yet concise presentation of the conceptual framework given above including a useful generalization for the case of reduction relations on quotient sets is given in Huet [55]. It is based on various earlier contributions, for instance Aho, Sethi, Ullman [1], Rosen [120], Lankford [70], Sethi [124], Staples [129].

It turns out that, implicitly, the idea of critical pair and completion is at the beginning of algorithmic mathematics: Euclid's algorithm and Gauss' algorithm may be viewed as critical pair/completion (cpc-) algorithms (see Section 11). The algorithm in Buchberger [12] for constructing canonical bases for polynomial ideals (see Section 11) seems to be the first one that explicitly stated the cpc-method, although several older algebraic algorithms (for instance, in group theory) have much of the flavor of cpc, for instance Dehn [28], Todd-Coxeter [136], Evans [33]. Also the resolution algorithm of Robinson [117] for automated theorem proving might be considered as a cpc-algorithm. The most general cpc-algorithm in the context of computer algebra is the Knuth-Bendix algorithm for rewrite rules (see Sections 9 and 10), whose forerunner, in fact, is Evans [33]. Later algorithms of the cpc type are the collection algorithm of McDonald [91], Newman [99] in computational group theory (see the respective chapter in this volume) and the algorithm of Bergman [5] for associative k-algebras. In Loos [81] some of these algorithms are analyzed under the cpc point of view and ideas for conceiving them as special cases of the Knuth-Bendix algorithms are presented.

9. The Knuth-Bendix Critical Pair Algorithm

The Knuth-Bendix critical pair algorithm is intended to yield a solution to the simplification problem for equivalence relations of the form $=_E$ (see Section 2) on sets of terms, where E is a set of equations in Term(X, a). For an exact formulation of the algorithm some additional notions for describing the replacement of terms in terms are needed. We explain these notions in an example (a formal definition may be based on a formalization of the concept of "tree", see for instance O'Donnell [101]):

Example. The term $t := gfxygxy3x$ $(a(f) = 2, a(g) = 3)$ has the tree representation shown in Fig. 2.

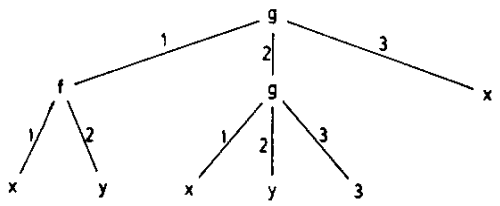

Fig. 2

One says that *fxy occurs* at the place (1) in t, $gxy3$ occurs at the place (2) in t, y occurs at the place $(1,2)$ in t etc., in symbols: $t/(1) = fxy$, $t/(2) = gxy3$, $t/(1,2) = y$ etc. In addition, $t/\Lambda = t$ where Λ is the empty sequence of natural numbers. The set of *occurrences* ("addresses", "places") $\mathrm{Oc}(t)$ of t in the above example is $\{\Lambda, (1), (2), (3), (1,1), (1,2), (2,1), (2,2), (2,3)\}$. ■

Let $\cdot$ denote concatenation on the set $\mathbb{N}^*$ of occurrences. The *prefix ordering* $\leqslant$ on $\mathbb{N}^*$ is defined by: $u \leqslant v$ iff $u \cdot w = v$ for some w. Furthermore, $v/u := w$ if $u \cdot w = v$, and $u \mid v$ (u and v are *disjoint*) iff neither $u \leqslant v$ nor $v \leqslant u$.

Finally, for terms s, t and occurrences u, $t[u \leftarrow s]$ is the term that derives from t if the term occurring at u in t is *replaced* by the term s. For instance, in the above example $t[(2) \leftarrow ffxx2] = gfxyffxx2x$.

Now, let E be a set of equations on $\mathrm{Term}(X, a)$. In a natural way, a reduction relation $\rightarrow_E$ on Term may be defined such that $\leftrightarrow^*_E = =_E$. Roughly, $s \rightarrow_E t$ iff t derives from s by applying one of the equations in E as "*rewrite rule*", i.e. in a directed way from left to right.

Definition. $s \rightarrow_E t$ (s *reduces* to t by E) iff there is a rule $(a, b) \in E$, a substitution σ and an occurrence $u \in \mathrm{Oc}(s)$ such that

$$s/u = \sigma(a) \qquad \text{and} \qquad t = s[u \leftarrow \sigma(b)]. \quad \blacksquare$$

Example. Let E be the following axiom system for group theory (in infix notation): (1) $1 \cdot x = x$, (2) $x^{-1} \cdot x = 1$, (3) $(x \cdot y) \cdot z = x \cdot (y \cdot z)$. Then, $(x^{-1} \cdot x) \cdot z \rightarrow_E 1 \cdot z \rightarrow_E z$. ■

In the sequel, we admit only those E, which satisfy: variable set of $b \subseteq$ variable set of a (for all $(a, b) \in E$). The next definition is crucial.

Definition. The terms p and q form a *critical pair* in E iff there are rules (a_1, b_1) and (a_2, b_2) in E and an occurrence u in $\mathrm{Oc}(a_1)$ such that

a_1/u is not a variable,
a_1/u and a_2 are unifiable,
$p = \sigma_1(a_1)[u \leftarrow \sigma_2(b_2)]$ and $q = \sigma_1(b_1)$,

where σ_1, σ_2 are substitutions such that

$\sigma_1(a_1/u) = \sigma_2(a_2)$ is a most general common instance of a_1/u and a_2 (with the property that no variable of $\sigma_1(a_1/u)$ occurs in a_1)

(i.e. p and q result from applying the rules (a_1, b_1) and (a_2, b_2) to a "most general match" of a_1 and a_2). ■

In the above definition, the notions of a "most general common instance" and of "unifiable terms" have been used. Let s, t_1, t_2 be terms. t_1 is an *instance* of t_2 iff $t_1 = \sigma(t_2)$ for some substitution σ. s is a *common instance* of t_1 and t_2 iff s is an instance of t_1 and t_2. t_1 and t_2 are *unifiable* iff t_1 and t_2 have a common instance. s is a *most general common instance* of t_1 and t_2 iff s is a common instance of t_1 and t_2 and every common instance of t_1 and t_2 is an instance of s. There is a straightforward algorithm ("*unification algorithm*", Robinson [117]) that decides for given t_1 and t_2 whether t_1 and t_2 are unifiable and, in the positive case, finds a most general common instance s of t_1 and t_2 (which is unique up to "permutations" of the variables) and substitutions σ_1 and σ_2 such that $s = \sigma_1(t_1) = \sigma_2(t_2)$. More sophisticated unification algorithms have been developed, for instance in Robinson [118], Paterson, Wegman [103], Huet [53], Martelli, Montanari [86].

Example. Consider rules (3) and (2) in the above example and the subterm $x \cdot y$ in (3), i.e. consider $a_1 = (x \cdot y) \cdot z$, $b_1 = x \cdot (y \cdot z)$, $a_2 = x^{-1} \cdot x$, $b_2 = 1$, and take $u = (1)$. $x^{-1} \cdot x$ is a most general instance of $a_1/u = x \cdot y$ and $a_2 = x^{-1} \cdot x$ (such that the condition on the variables is satisfied; take $\sigma_1(x) = x^{-1}$, $\sigma_1(y) = x$, $\sigma_2 =$ identity). Hence, $p = \sigma_1(a_1)[u \leftarrow \sigma_2(b_2)] = (x^{-1} \cdot x) \cdot z[(1) \leftarrow 1] = 1 \cdot z$ and $q = \sigma_1(b_1) = x^{-1} \cdot (x \cdot z)$ form a critical pair in E. ■

Theorem (Reduction of local confluence to confluence of critical pairs. Knuth-Bendix 1967). *$\rightarrow_E$ is locally confluent iff for all critical pairs (p, q) of E: $p \downarrow_E q$.* ■

If E is finite, one always can construct an algorithm Sel such that $t \rightarrow_E \mathrm{Sel}(t)$ if t is not in normal form. If $\rightarrow_E$ is noetherian, let S_E be the normal-form algorithm based on Sel.

Algorithm (Critical pair algorithm for rewrite rules, Knuth-Bendix 1967).

Input: E (a set of equations).

Question: Is the normal-form algorithm S_E a canonical simplifier for $=_E$?

$C :=$ set of critical pairs of (E)
for all $(p, q) \in C$ *do*
  $(p_0, q_0) := (S_E(p), S_E(q))$
  if $p_0 \neq q_0$ *then* answer: "S_E is not canonical".
answer: "S_E is canonical". ■

The correctness of the algorithm is an easy consequence of the above theorem and the lemmas in the preceding section.

Proof of the Theorem (sketch). "⇒": In the notation of the above definition, $p \leftarrow \sigma_1(a_1) \rightarrow q$ for critical pairs (p, q). (For simplicity, we write $\rightarrow$ instead of $\rightarrow_E$ etc.). Hence, $p \downarrow q$ by local confluence.

"⇐": Let t_1, s, t_2 be arbitrary terms and assume $t_1 \leftarrow s \rightarrow t_2$. We have to show $t_1 \downarrow t_2$, i.e. $t_1 \rightarrow^* w \leftarrow^* t_2$ for some w. By assumption, there are rules (a_1, b_1) and (a_2, b_2) in E, occurrences u_1, u_2 in $\mathrm{Oc}(s)$, and substitutions σ_1, σ_2 such that $s/u_1 = \sigma_1(a_1)$, $s/u_2 = \sigma_2(a_2)$, $t_1 = s[u_1 \leftarrow \sigma_1(b_1)]$, $t_2 = s[u_2 \leftarrow \sigma_2(b_2)]$. There are essentially the two cases shown in Fig. 3:

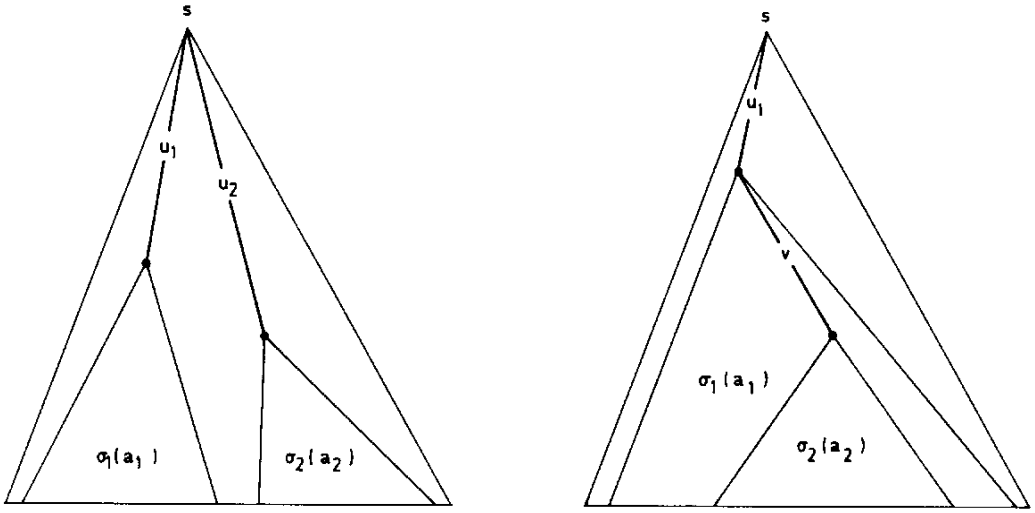

Fig. 3

(The case $u_2 \cdot v = u_1$ is symmetric. Drawings in the above style may be very helpful for clarifying the subsequent arguments).

In the first case: $t_1/u_2 = s[u_1 \leftarrow \sigma_1(b_1)]/u_2 = s/u_2 = \sigma_2(a_2)$, $t_2/u_1 = \sigma_1(a_1)$. Define $w := t_1[u_2 \leftarrow \sigma_2(b_2)]$. Then $t_1 \to w$ and, also, $t_2 \to w$, because

$$w = s[u_1 \leftarrow \sigma_1(b_1)][u_2 \leftarrow \sigma(b_2)] = s[u_2 \leftarrow \sigma_2(b_2)][u_1 \leftarrow \sigma_1(b_1)] =$$
$$= t_2[u_1 \leftarrow \sigma_1(b_1)].$$

In the second case: $t_1 = s[u_1 \leftarrow \sigma_1(b_1)]$, $t_2 = s[u_1 \leftarrow \sigma_1(a_1)[v \leftarrow \sigma_2(b_2)]]$. If one can show that there exists w_0 such that $\sigma_1(b_1) \to^* w_0 \leftarrow^* \sigma_1(a_1)[v \leftarrow \sigma_2(b_2)]$, then also $t_1 \downarrow t_2$ by the *compatibility* property of $\to_E$ (see definition below).

There are the two subcases shown in Fig. 4.

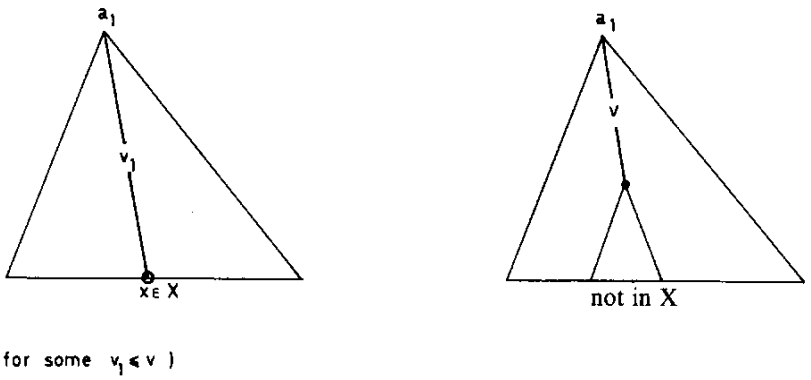

Fig. 4

In the first subcase: $\sigma_2(a_2) = \sigma_1(x)/v_2$, where v_2 is such that $v_1 \cdot v_2 = v$. Define a new substitution σ_1' by $\sigma_1'(x) := \sigma_1(x)[v_2 \leftarrow \sigma_2(b_2)]$, $\sigma_1'(y) := \sigma_1(y)$ for $y \neq x$, and

define $w_0 := \sigma'_1(b_1)$. Then $\sigma_1(b_1) \to^* \sigma'_1(b_1)$, because $\sigma_1(x) \to \sigma'_1(x)$. Also, $\sigma_1(a_1)[v \leftarrow \sigma_2(b_2)] = \sigma_1(a_1)[v_1 \leftarrow \sigma'_1(x)] \to^* \sigma'_1(a_1)$. Since $a_1 \to b_1$ we also have $\sigma'_1(a_1) \to \sigma'_1(b_1)$ ($\to_E$ is *stable*, see definition below). Hence, $\sigma_1(b_1) \to^* w_0 \leftarrow^* \sigma_1(a_1)[v \leftarrow \sigma_2(b_2)]$.

In the second subcase it is not difficult to show that the situation considered is an instance of a *critical pair*, i.e. there is a critical pair (p, q) of E and a substitution ρ such that $\sigma_1(a_1)[v \leftarrow \sigma_2(b_2)] = \rho(p)$ and $\sigma_1(b_1) = \rho(q)$. By assumption there is an r such that $p \to^* r \leftarrow^* q$. Define $w_0 := \rho(r)$. Again by the *stability* of $\to_E$ we have $\sigma_1(b_1) \to^* w_0 \leftarrow^* \sigma_1(a_1)[v \leftarrow \sigma_2(b_2)]$. ■

For providing the details of the above proof a number of lemmas must be proven, which are pictorially evident (however, pictures may be misleading!) but tedious. (An exact recursive definition of the basic notions as, for instance, replacement, occurrence etc. would be necessary.) These lemmas have been tacitly used in the above sketch. One example of such a lemma is the "*commutativity* of replacement":

$$u_1 | u_2 \Rightarrow s[u_1 \leftarrow t_1][u_2 \leftarrow t_2] = s[u_2 \leftarrow t_2][u_1 \leftarrow t_1].$$

Two important lemmas, which have been mentioned explicitly in the above proof, are:

Lemma.

$\to_E$ *is stable*, *i.e. for all terms* s, t *and substitutions* σ:

$$(s \to_E t \Rightarrow \sigma(s) \to_E \sigma(t)).$$

$\to_E$ *is compatible*, *i.e. for all terms* s, t_1, t_2 *and* $u \in \mathrm{Oc}(s)$:

$$(t_1 \to_E t_2 \Rightarrow s[u \leftarrow t_1] \to_E s[u \leftarrow t_2]).$$ ■

Example. In the above example of group theory the critical pair (p, q) resulting from rules (3) and (2) has no common successor. The only reduction possible for p is $p = 1 \cdot z \to z$ and $q = x^{-1} \cdot (x \cdot z)$ is already in normal form. Hence, $\to_E$ is not locally confluent and the normal-form algorithm S_E is not canonical.

Consider now the following axiom system E' for group theory (Knuth-Bendix [65]), which results from the above system by adding some identities that are theorems in group theory:

(1) $1 \cdot x = x$, (2) $x^{-1} \cdot x = 1$, (3) $(x \cdot y) \cdot z = x \cdot (y \cdot z)$,

(4) $1^{-1} = 1$, (5) $x^{-1} \cdot (x \cdot y) = y$, (6) $x \cdot 1 = x$,

(7) $(x^{-1})^{-1} = x$, (8) $x \cdot x^{-1} = 1$, (9) $x \cdot (x^{-1} \cdot y) = y$,

(10) $(x \cdot y)^{-1} = y^{-1} \cdot x^{-1}$.

It can be shown (Knuth-Bendix [65]) that $\to_{E'}$ is noetherian. Furthermore, by finitely many checks it can be seen that $S_{E'}(p) = S_{E'}(q)$ for all critical pairs of E'. In particular, the above critical pair has now the common successor z. Hence, by the theorem $\to_{E'}$ is (locally) confluent, $S_{E'}$ is a canonical simplifier for $\leftrightarrow^*_{E'} = \leftrightarrow^*_E = =_E$. This means that the decision about the validity of identities in group theory can be fully automatized. ■

For applying the Knuth-Bendix algorithm to a set E of equations $\rightarrow_E$ must be proven noetherian. In general this is a non-trivial task that needs specific methods for each example. The uniform halting problem, i.e. the question whether there exists an infinite chain $t \rightarrow_E t_1 \rightarrow_E t_2 \rightarrow_E \cdots$ (t and E free parameters) is undecidable, see Huet, Lankford [57]). On the other hand, the uniform halting problem for rewrite system E that consist entirely of ground terms is decidable, see Huet, Lankford [57]. A number of important techniques have been developed for proving $\rightarrow_E$ noetherian, for instance Knuth, Bendix [65], Manna, Ness [85], Lankford [69, 70], Plaisted [106, 107], Dershowitz [29, 30], Dershowitz, Manna [31], see the survey in Huet, Oppen [58]. Basically, these techniques proceed from a well-founded partial ordering $>$ on the set of ground terms. It then is shown that $s \rightarrow_E t \Rightarrow s > t$ (for ground terms s, t). This suffices for proving $\rightarrow_E$ noetherian because it is known that $\rightarrow_E$ is noetherian iff there is no ground term having an infinite sequence of reductions.

Important modifications of the Knuth-Bendix (critical pair and completion algorithm) have been given in Plotkin [108], Lankford, Ballantyne [73, 74, 75], Huet [55], and Peterson, Stickel [105]. Mainly, these modifications consist in considering confluence modulo equivalence relations generated by standard axioms (as, for instance, commutativity), which do not naturally suggest a distinction between left-hand and right-hand side. Unification modulo such sets of axioms plays an essential role in this context, see the surveys of Raulefs, Siekmann, Szabo, Unvericht [111] and Huet, Oppen [58].

10. The Knuth-Bendix Completion Algorithm

If the Knuth-Bendix critical pair algorithm for a set E of equations shows that the reduction of a critical pair (p, q) yields $S(p) \neq S(q)$ then it suggests itself to try to augment E by the rule $(S(p), S(q))$ (or $(S(q), S(p))$) in order to achieve completeness: in fact, the reflexive-symmetric-transitive closure of $\rightarrow_E$ is not changed by this adjunction, whereas $\rightarrow_E$ itself is properly extended. This process may be iterated until, hopefully, "saturation" will be reached, i.e. all critical pairs have a unique normal form. In this case the final set of equations still generates the same equivalence relation $=_E$ as the original one, but its reduction relation $\rightarrow_E$ is unique, i.e. the associated normal form algorithm is a canonical simplifier for $=_E$ (and hence, $=_E$ is decidable). Of course, before extending E one must check whether $\rightarrow_E$ remains noetherian by this extension and in order to meet this condition one must choose between the two possibilities $(S(p), S(q))$ and $(S(q), S(p))$. It may also happen that the preservation of the finite termination property cannot be proven for either possibility. In this case the completion process cannot be continued reasonably. We present the rough structure of this procedure:

Algorithm (Completion algorithm for rewrite rules. Knuth-Bendix 1967).

Given: E, a finite set of equations such that $\rightarrow_E$ is noetherian.
Find: F, a finite set of equations such that
$\leftrightarrow^*_E = \leftrightarrow^*_F$ and
$\rightarrow_F$ is (locally) confluent ($\Leftrightarrow S_F$ is a canonical simplifier).

$F := E$
$C :=$ set of critical pairs of (F)

while $C \neq 0$ *do*
  $(p, q) :=$ an element in (C)
  $(p_0, q_0) := (S_F(p), S_F(q))$
  if $p_0 \neq q_0$ *then*
    Analyze (p_0, q_0)
    $C := C \cup$ set of new critical pairs $((p_0, q_0), F)$
    $F := F \cup \{(p_0, q_0)\}$
  $C := C - \{(p, q)\}$

stop with success. ■

The subroutines "set of critical pairs of" and "an element in" seem to be self-explanatory. The subroutine "set of new critical pairs" computes the critical pairs deriving from the new rule (p_0, q_0) and the rules in F. The subroutine "Analyze" determines whether $\rightarrow_E$ remains noetherian when (p_0, q_0) or (q_0, p_0) is added to E. In the first case (p_0, q_0) is left unchanged, in the second case the roles of p_0 and q_0 are interchanged. If none of the two alternatives holds, the subroutine "Analyze" executes a *stop* with failure. There are three possibilities: 1. The algorithm stops with success. In this case the final F meets the specifications stated in the heading of the algorithm. 2. The algorithm stops with failure. In this case nothing interesting can be said. 3. The algorithm never stops: In this case the algorithm is at least a semidecision procedure for $=_E$. It can be shown (Huet [54]) that $s =_E t$ can be semidecided by reducing s and t with respect to the steadily increasing set of equations produced by the algorithm.

The above crude form of the algorithm can be organizationally refined in many ways. For instance, the two sides of an equation in F can be kept in reduced form relative to the other equations in F. Furthermore, the critical pairs can be generated in an "incremental way" (Huet [54]). The sequence of critical pairs chosen by the procedure "an element in" may have a crucial influence on the efficiency of the algorithm. These questions are subtle.

Implementations of the Knuth-Bendix completion algorithm are reported in Knuth, Bendix [65], Hullot [59], Richter, Kemmenich [115], Loos [80], Peterson, Stickel [105]. Investigations of the *complexity* of the algorithm are difficult. Necessarily they are confined to special applications of the algorithm. A survey on known complexity results is given in Lankford [71]. An analysis of various subalgorithms used in the Knuth-Bendix algorithm may be found in Loos [80].

An impressive variety of *axiom systems* have been successfully completed by the Knuth-Bendix algorithm (Knuth, Bendix [65], Lankford, Ballantyne [73, 74, 75], Stickel, Peterson [130], Ballantyne, Lankford [4], Huet, Hullot [56], Hullot [59], Bücken [14], Richter, Kemmenich [115]), including axiom systems for groups (see example in the last section), central groupoids, loops, (l, r) systems, commutative rings with 1, R-modules, fragments of number theory, distributive lattices, non-deterministic machines, "robot" theory, special finitely presented groups and commutative semigroups. Recently, the Knuth-Bendix completion algorithm is

extensively used for completing the defining equations of *abstract data types*, for instance in the AFFIRM system (Gerhart et al. [42], Musser [97]). By the theorem on canonical simplification and computing in algebraic domains (see Section 1) completed axiom systems allow to effectively compute in the "direct implementations of the abstract data types", i.e. in the term algebra modulo the equivalence generated by the equations (Musser [97], Lichtenberger [78], Kapur, Musser, Stepanov [62]). Examples of complete equational specifications of abstract data types are given in Musser [97, 98].

Recently (Courcelle [24], Musser [98], Goguen [43], Huet, Hullot [56]), a new type of application of the completion process has been proposed: completion as a substitute for induction, see also Lankford [72]. By this method, the validity of equations e in the "initial model" of a set E of equations (Goguen et al. [15, 44]) may be proven by testing whether the Knuth-Bendix completion algorithm applied to $E \cup \{e\}$ stops without generating any inconsistency.

11. A Completion Algorithm for Polynomial Ideal Bases

Let E be a (finite) set of *polynomials* in $K[x_1, \ldots, x_n]$, K a field and let $<_T$ be a linear *ordering of the monomials*, which is "*admissible*" in the sense that $\sigma_1 <_T \sigma_2 \Rightarrow \tau \cdot \sigma_1 <_T \tau \cdot \sigma_2$ and $1 \leqslant_T \sigma$ (τ, σ_1, σ_2 monomials). We study the problem of constructing a canonical simplifier for the *congruence relation* $\equiv_E$ modulo the polynomial ideal Ideal(E) generated by the polynomials in E. It is easy (although non-trivial) to show that the following reduction relation $\rightarrow_E$ is such that $\leftrightarrow^*_E = \equiv_E$:

Definition. $s \rightarrow_E t$ (s *reduces to* t modulo E) iff there is a polynomial (a, b) in E, a monomial σ and a monomial u occurring in s such that

$$u = \sigma \cdot a \qquad \text{and} \qquad t = s[u \leftarrow \sigma \cdot b]. \quad \blacksquare$$

The polynomials of E are thought to be represented in E in the form (a, b), where a is the leading monomial in the polynomial with respect to $<_T$ with its leading coefficient normed to 1 and b is the rest of the polynomial. Structurally, this definition is very similar to the definition of $\rightarrow_E$ in the preceding section. The multiplication with a monomial corresponds to the application of a substitution and the operation of replacement in the present context is defined as follows:

$s[u \leftarrow t] :=$ the polynomial that results from the polynomial s by replacing the monomial u by the polynomial t.

Example.

$$E := \{(x^2y, -3y^2 + 2y), (y^2, -2x)\}.$$

$$s := x^4 - 2x^2y^2 + 5y^3 + x \rightarrow_E x^4 - 11y^3 - 4y^2 + x =: t,$$

because the monomial x^2y^2 occurring in s is a multiple $u \cdot x^2y$ of the left-hand side of the first polynomial in E, where u is the monomial y. t results from s by replacing x^2y^2 by $u \cdot (-3y^2 + 2y)$. $\blacksquare$

It is not difficult to show that $\rightarrow_E$ is noetherian for arbitrary E. It is clear that there exists a selector function Sel_E and, based on Sel_E, a normal form algorithm S_E

for $\rightarrow_E$. Again, the following notion is basic for the approach to constructive ideal theory given here.

Definition. The polynomials p and q form a *critical pair* of E iff there are polynomials (a_1, b_1) and (a_2, b_2) in E such that

$$p = \sigma_1(b_1) \quad \text{and} \quad q = \sigma_2(b_2)$$

where σ_1, σ_2 are monomials such that

$$\sigma_1 \cdot a_1 = \sigma_2 \cdot a_2 \text{ is the least common multiple of } a_1 \text{ and } a_2. \quad \blacksquare$$

Example.

$$E := \{(x^3yz, xz^2), (xy^2z, xyz), (x^2y^2, z^2)\}.$$

x^3y^2z is a least common multiple of the left-hand sides of the first two polynomials in E. $\sigma_1 = y$, $\sigma_2 = x^2$. $p = xyz^2$, $q = x^3yz$. $\blacksquare$

Again, one can show that the consideration of the critical pairs is sufficient for deciding the (local) confluence of $\rightarrow_E$:

Theorem (Critical pair algorithm for polynomial ideals. Buchberger 1965). *S_E is a canonical simplifier for $\rightarrow_E$ iff*

$$\textit{for all critical pairs of } (p, q) \textit{ of } E\text{: } S_E(p) = S_E(q). \quad \blacksquare$$

Proof. Again, the proof of this theorem may be organized in such a way that, essentially it consists in the proof of: $\rightarrow_E$ is locally confluent iff for all critical pairs (p, q) of E: $p \downarrow q$. At present, no proof of this property is known which would proceed by a specialization of the Knuth-Bendix proof. A special proof is necessary, see Buchberger [12, 13], Bachmair, Buchberger [3]. The main reason for this seems to be the fact that $\rightarrow_E$ is not compatible, but only "semi-compatible":

$\rightarrow_E$ is *stable*, i.e. for all polynomials s, t and monomials σ:

$$(s \rightarrow_E t \Rightarrow \sigma \cdot s \rightarrow_E \sigma \cdot t).$$

$\rightarrow_E$ is *semi-compatible*, i.e. for all polynomials s, t_1, t_2 and monomials u occurring in s:

$$(t_1 \rightarrow_E t_2 \Rightarrow s[u \leftarrow t_1] \downarrow_E s[u \leftarrow t_2]). \quad \blacksquare$$

Example. In the above example $S_E(p) = \underline{xyz^2}$, $S_E(q) = \underline{xz^2}$. Hence, by the theorem S_E is not canonical. $\blacksquare$

Again the critical pair algorithm can naturally be modified to obtain the following completion algorithm:

Algorithm (Completion algorithm for polynomial ideal bases. Buchberger 1965).

Given: E, a finite set of polynomials in $K[x_1, \ldots, x_n]$.
Find: F, a finite set of polynomials in $K[x_1, \ldots, x_n]$ such that
$\equiv_E = \equiv_F$ and
$\rightarrow_F$ is (locally) confluent ($\Leftrightarrow S_F$ is a canonical simplifier)
(An F with this property is called *Gröbner* basis).

The algorithm is formally identical to the Knuth-Bendix completion algorithm, but with different meaning associated to the subroutines. In particular the subroutine "Analyze" has no *stop* with failure. Instead "Analyze" performs $(p_0, q_0) :=$ leading monomial and rest of the polynomial $p_0 - q_0$ (after normalizing the leading coefficient to 1). ■

The algorithm stops for arbitrary input sets E, see Buchberger [12], and Bergman [5], p. 208. Again, a number of organizational improvements are possible for the algorithm, the most important being the one developed in Buchberger [13], which shows that, instead of critical pairs, "chains" of critical pairs may be considered. This results in a drastic improvement in the speed of the algorithm. This approach might also prove fruitful for other instances of cpc algorithms. Research in this direction has now been undertaken.

In the special case of $E \subseteq K[x]$ the above algorithm specializes to Euclid's algorithm. In the special case of a set of linear polynomials in $K[x_1, \ldots, x_n]$ the algorithm specializes to Gauss' algorithm. In the special case of having only monomials on the right-hand sides of the polynomials in E, E are the defining relations of a finitely generated commutative semigroup and the algorithm is a decision procedure for the uniform word problem for commutative semigroups. It has been shown by Cardoza, Lipton, Meyer [16], and Mayr, Meyer [90] that this problem is complete in exponential space under log-space transformability. Hence, also the complexity of the above algorithm necessarily will be high. Nevertheless, in practical examples it has proven feasible. An explicit complexity bound for the case $n = 2$ has been derived in Buchberger [13]. Various implementations of the algorithm have been reported, see Buchberger [12], Schrader [122], Lauer [76], Spear [128], Trinks [137], Zacharias [141], Schaller [121], Winkler et al. [139], including generalizations for polynomials over rings and various applications (effective computation in residue rings modulo polynomial ideals, computation of elimination ideals and solution of sets of algebraic equations, effective Lasker-Noether decomposition of polynomial ideals, effective computation of the Hilbert function and the sycygies of polynomial ideals, interpolation formulae for numerical cubature (Möller [92])). A generalization of the algorithm for associative algebras with many interesting applications (for instance for Lie-algebras) over commutative rings with 1 has independently been derived in Bergman [5].

For improving the computational feasibility of the algorithm the application of the techniques derived for Euclid's algorithm (Collins [23], Brown [10]; see the chapter on polynomial remainder sequences) should be investigated. A comparison of the notion of resultant, polynomial remainder sequence and reduction $\rightarrow_E$ is necessary for this purpose. The first observations in this direction appear in Schaller [121], Pohst, Yun [109]. Also a comparison with other approaches for deriving "canonical bases" for polynomial ideals and constructive methods in ring theory (Hermann [50], Szekeres [134], Kleiman [64], Seidenberg [123], Shtokhamer [125], Trotter [138], Ayub [2]), seems to be promising.

References

[1] Aho, A., Sethi, R., Ullman, J.: Code Optimization and Finite Church-Rosser Systems. Proc. of Courant Comput. Sci. Symp. 5 (Rustin, R., ed.). Englewood Cliffs, N. J.: Prentice-Hall 1972.

[2] Ayoub, C. W.: On Constructing Bases for Ideals in Polynomial Rings over the Integers. Pennsylvania State University, Univ. Park: Dept. Math. Rep. **8184** (1981).
[3] Bachmair, L., Buchberger, B.: A Simplified Proof of the Characterization Theorem for Gröbner Bases. ACM SIGSAM Bull. **14**/4, 29–34 (1980).
[4] Ballantyne, A. M., Lankford, D. S.: New Decision Algorithms for Finitely Presented Commutative Semigroups. Comp. Math. Appl. **7**, 159–165 (1981).
[5] Bergman, G. M.: The Diamond Lemma for Ring Theory. Adv. Math. **29**, 178–218 (1978).
[6] Birkhoff, G.: On the Structure of Abstract Algebras. Proc. Cambridge Phil. Soc. **31**, 433–454 (1935).
[7] Birkhoff, G., Lipson, J. D.: Heterogeneous Algebras. J. Comb. Theory **8**, 115–133 (1970).
[8] Boyer, R. S., Moore, J. S.: A Computational Logic. New York-London: Academic Press 1979.
[9] Brown, W. S.: Rational Exponential Expressions and a Conjecture Concerning π and e. Am. Math. Mon. **76**, 28–34 (1969).
[10] Brown, W. S.: On Euclid's Algorithm and the Computation on Polynomial Greatest Common Divisor. J. ACM **18**/4, 478–504 (1971).
[11] Brown, W. S.: On Computing with Factored Rational Expressions. EUROSAM **1974**, 26–34.
[12] Buchberger, B.: An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (German). Ph.D. Thesis, Math. Inst., Univ. of Innsbruck, Austria, 1965, and Aequationes Math. **4**/3, 374–383 (1970).
[13] Buchberger, B.: A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. EUROSAM **1979**, Lecture Notes in Computer Science, Vol. 72, pp. 3–21. Berlin-Heidelberg-New York: Springer 1979.
[14] Bücken, H.: Reduktionssysteme und Wortproblem. Rhein.-Westf. Tech. Hochschule, Aachen: Inst. für Informatik, Rep. **3**, 1979.
[15] Burstall, R. M., Goguen, J. A.: Putting Theories Together to Make Specifications. Proc. 5th Internat. Joint Conf. on Artificial Intelligence **1977**, 1045–1058.
[16] Cardoza, E., Lipton, R., Meyer, A. R.: Exponential Space Complete Problems for Petri Nets and Commutative Semigroups. Conf. Record of the 8th Annual ACM Symp. on Theory of Computing, 50–54 (1976).
[17] Caviness, B. F.: On Canonical Forms and Simplification, Ph.D. Diss., Pittsburgh, Carnegie-Mellon University, 1967, and J. ACM **17**/2, 385–396 (1970).
[18] Caviness, B. F., Fateman, R.: Simplification of Radical Expressions. SYMSAC **1976**, 329–338.
[19] Caviness, B. F., Pollack, P. L., Rubald, C. M.: An Existence Lemma for Canonical Forms in Symbolic Mathematics. Inf. Process. Lett. **1**, 45–46 (1971).
[20] Church, A., Rosser, J. B.: Some Properties of Conversion. Trans. AMS **39**, 472–482 (1936).
[21] Cohn, P. M.: Universal Algebra. New York: Harper and Row 1965.
[22] Collins, G. E.: The SAC-1 Polynomial System. Univ. of Wisconsin, Madison: Comput. Sci. Dept., Tech. Rep. **115**, 1968.
[23] Collins, G. E.: The Calculation of Multivariate Polynomial Resultants. J. ACM **18**/4, 515–532 (1971).
[24] Courcelle, B.: On Recursive Equations Having a Unique Solution. IRIA – Laboria Rep. **285** (1976).
[25] Davenport, J.: On the Integration of Algebraic Functions. Univ. of Cambridge: Ph.D. Thesis. (Lecture Notes in Computer Science, Vol. 102.) Berlin-Heidelberg-New York: Springer 1981.
[26] Davis, M.: Hilbert's Tenth Problem is Unsolvable. Am. Math. Mon. **80**/3, 233–269 (1973).
[27] Davis, M., Putnam, H., Robinson, J.: The Decision Problem for Exponential Diophantine Equations. Ann. Math. **74**, 425–436 (1961).
[28] Dehn, M.: Über unendliche diskontinuierliche Gruppen. Math. Ann. **71**, 116–144 (1911).
[29] Dershowitz, N.: A Note on Simplification Orderings. Inf. Process. Lett. **9**/5, 212–215 (1979).
[30] Dershowitz, N.: Orderings for Term-Rewriting Systems. Proc. 20th Symp. on Foundations of Comp. Sci. **1979**, 123–131.
[31] Dershowitz, N., Manna, Z.: Proving Termination with Multiset Ordering. Commun. ACM **22**, 465–476 (1979).
[32] Epstein, H. I.: Using Basis Computation to Determine Pseudo-Multiplicative Independence. SYMSAC **1976**, 229–237.
[33] Evans, T.: The Word Problem for Abstract Algebras. J. London Math. Soc. **26**, 64–71 (1951).

[34] Fateman, R. J.: The User-Level Semantic Pattern Matching Capability in MACSYMA. Proc. SYMSAM **1971**, 311–323.
[35] Fateman, R. J.: Essays in Algebraic Simplification. Thesis, MIT, Project MAC 1972.
[36] Fateman, R. J.: MACSYMA's General Simplifier: Philosophy and Operation. MACSYMA **1979**, 563–582.
[37] Fateman, R. J.: Symbolic and Algebraic Computer Programming Systems. ACM SIGSAM Bull. **15**/1, 21–32 (1981).
[38] Fenichel, J.: An One-Line System for Algebraic Manipulation. Harvard Univ., Cambridge, Mass.: Ph.D. Thesis 1966.
[39] Fitch, J. P.: An Algebraic Manipulator. Ph.D. Thesis, Univ. of Cambridge, 1971.
[40] Fitch, J. P.: On Algebraic Simplification. Comput. J. **16**/1, 23–27 (1973).
[41] Foderaro, J. K.: Typesetting MACSYMA Equations. MACSYMA **1979**, 345–361.
[42] Gerhart, S. L., Musser, D. R., Thompson, D. H., Baker, D. A., Bates, R. L., Erickson, R. W., London, R. L., Taylor, D. G., Wile, D. S.: An Overview of AFFIRM: A Specification and Verification System. In: Information Processing 80 (Lavington, S. H., ed.), pp. 343–347. Amsterdam: North-Holland 1980.
[43] Goguen, J. A.: How to Prove Algebraic Inductive Hypotheses without Induction, with Applications to the Correctness of Data Type Implementations. Proc. 5th Conf. on Aut. Deduction. Lecture Notes in Computer Science, Vol. 87, pp. 356–373. Berlin-Heidelberg-New York: Springer 1980.
[44] Goguen, J. A., Thatcher, J. W., Wagner, E. G., Wright, J. B.: Abstract Data Types as Initial Algebras and Correctness of Data Representations. Proc. Conf. on Comput. Graphics, Pattern Recognition and Data Structure **1975**, 89–93, Beverly Hills, California.
[45] Griss, M. L: The Definition and Use of Datastructures in REDUCE. SYMSAC **1976**, 53–59.
[46] Hall, A. D.: Factored Rational Expressions in ALTRAN. EUROSAM **1974**, 35–45.
[47] Hearn, A. C.: A New REDUCE Model for Algebraic Simplification. SYMSAC **1976**, 46–52.
[48] Hearn, A. C.: Symbolic Computation: Past, Present, Future. Dept. of Comput. Sci., Univ. Utah, Salt Lake City 1978.
[49] Henrici, P.: A Subroutine for Computations with Rational Numbers. J. ACM **3**/1, 6–9 (1956).
[50] Hermann, G.: Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. Math. Ann. **95**, 736–788 (1926).
[51] Hindley, R.: An Abstract Form of the Church-Rosser Theorem II: Applications. J. Symb. Logic **39**/1, 1–21 (1974).
[52] Hindley, R., Lercher, B., Seldin, J. P.: Introduction to Combinatory Logic. Cambridge: Camb. Univ. Press 1972.
[53] Huet, G. P.: Résolution d'équations dans des languages d'ordre 1, 2, ..., ω. Ph.D. Thesis, University of Paris VII, 1976.
[54] Huet, G.: A Complete Proof of the Knuth-Bendix Competion Algorithm. IRIA: Report (1979).
[55] Huet, G. P.: Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. 18th IEEE Symp. on Foundat. of Comput. Sci., 30–45 (1977) and J. ACM **27**/4, 797–821 (1980).
[56] Huet, G. P., Hullot, J. M.: Proofs by Induction in Equational Theories with Constructors. 21st IEEE Symp. on Foundations of Comput. Sci. **1980**, 96–107.
[57] Huet, G., Lankford, D. S.: On the Uniform Halting Problem for Term Rewriting Systems. IRIA: Rapp. Laboria **283**, 1978.
[58] Huet, G. P., Oppen, D. C.: Equations and Rewrite Rules: A Survey. Techn. Rep. CSL-**111**, SRI International, Stanford, 1980.
[59] Hullot, J. M.: A Catalogue of Canonical Term Rewriting Systems. Stanford: SRI International Tech. Rep. CSL-**113**, 1980.
[60] Jenks, R. D.: A Pattern Compiler. SYMSAC **1976**, 60–65.
[61] Johnson, S. C.: On the Problem of Recognizing Zero. J. ACM **18**/4, 559–565 (1971).
[62] Kapur, D., Musser, D. R., Stepanov, A. A.: Operators and Algebraic Structures. Schenectady, N.Y.: General Electric Automation and Control Laboratory, Report 81CRD **114** (1981).
[63] King, J. C., Floyd, R. W.: An Interpretation Oriented Theorem Prover Over the Integers. J. Comput. Syst. Sci. **6**/4, 305–323 (1972).
[64] Kleiman, S. L.: Computing with Rational Expressions in Several Algebraically Dependent Variables. Columbia Univ., New York: Computing Science Techn. Report **42**, 1966.

[65] Knuth, D. E., Bendix, P. B.: Simple Word Problems in Universal Algebras. OXFORD 67, 263–298.
[66] Korsvold, K.: An On-Line Algebraic Simplify Program. Stanford Univ.: Art. Int. Project Memorandum **37**, 1965, and Comm. ACM **9**, 553 (1966).
[67] Lafferty, E. L.: Hypergeometric Function Reduction – an Adventure in Pattern Matching. MACSYMA **1979**, 465–481.
[68] Lang, S.: Transcendental Numbers and Diophantine Approximations. Bull. AMS **77**/5, 635–677 (1971).
[69] Lankford, D. S.: Canonical Algebraic Simplification. Univ. of Texas, Austin: Dept. Math. Comput. Sci., Rep. ATP-**25**, 1975.
[70] Lankford, D. S.: Canonical Inference. Univ. of Texas, Austin: Dept. Math. Comput. Sci., Rep. ATP-**32**, 1975.
[71] Lankford, D. S.: Research in Applied Equational Logic. Louisiana Tech. Univ., Ruston: Math. Dept., MTP-**15**, 1980.
[72] Lankford, D. S.: A Simple Explanation of Inductionless Induction. Louisiana Tech. Univ., Ruston: Math. Dept., MTP-**14**, 1981.
[73] Lankford, D. S., Ballantyne, A. M.: Decision Procedures for Simple Equational Theories with Commutative Axioms: Complete Sets of Commutative Reductions. Univ. of Texas, Austin: Dept. Math. Comput. Sci., Rep. ATP-**35**, 1977.
[74] Lankford, D. S., Ballantyne, A. M.: Decision Procedures for Simple Equational Theories with Permutative Axioms: Complete Sets of Permutative Reductions. Univ. of Texas, Austin: Dept. Math. Comput. Sci., Rep. ATP-**37**, 1977.
[75] Lankford, D. S., Ballantyne, A. M.: Decision Procedures for Simple Equational Theories with Commutative-Associative Axioms: Complete Sets of Commutative-Associative Reductions. Univ. of Texas, Austin: Dept. Math. Comput. Sci., Rep. ATP-**39**, 1977.
[76] Lauer, M.: Canonical Representatives for Residue Classes of a Polynomial Ideal. SYMSAC **1976**, 339–345.
[77] Lausch, H., Nöbauer, W.: Algebra of Polynomials. Amsterdam: North-Holland 1973.
[78] Lichtenberger, F.: PL/ADT: A System for Using Algebraically Specified Abstract Data Types in PL/I (German). Ph.D. Thesis, Math. Inst., Univ. of Linz, Austria, 1980.
[79] Loos, R.: Toward a Formal Implementation of Computer Algebra. EUROSAM **1974**, 9–16.
[80] Loos, R.: Algorithmen und Datenstrukturen I: Abstrakte Datentypen. Univ. Karlsruhe, Federal Republic of Germany: Lecture Notes, 1980.
[81] Loos, R.: Term Reduction Systems and Algebraic Algorithms. Proc. 5th GI Workshop on Artif. Intell., Bad Honnef 1981, Informatik Fachberichte **47**, 214–234 (1981).
[82] Loveland, D. W., Shostak, R. E.: Simplifying Interpreted Formulas. 5th Conf. on Automated Deduction, Les Arces, France, 1980. Lecture Notes in Computer Science, Vol. 87, pp. 97–109. Berlin-Heidelberg-New York: Springer 1980.
[83] Luckham, D. C., German, S. M., Henke, F. W., Karp, R. A., Milne, P. W., Oppen, D. C., Polak, W., Scherlis, W. L.: Stanford PASCAL Verifier User Manual. Stanford Univ.: Comput. Sci. Dept., Rep. STAN-CS-79-**731**, 1979.
[84] Manin, Y. I.: A Course in Mathematical Logic. Berlin-Heidelberg-New York: Springer 1977.
[85] Manna, Z., Ness, S.: On the Termination of Markov Algorithms. Third Hawaii International Conference on System Sciences **1970**, 789–792.
[86] Martelli, A., Montanari, U.: An Efficient Unification Algorithm. Unpublished Report (1979).
[87] Martin, W. A.: Computer Input/Output of Mathematical Expressions. SYMSAM **1971**, 78–89.
[88] Martin, W. A.: Determining the Equivalence of Algebraic Expressions by Hash Coding. SYMSAM **1971**, 305–310.
[89] Matiyasevic, J.: Diophantine Representation of Recursively Enumerable Predicates. Proc. Second Scandinavian Logic Symp. Amsterdam: North-Holland 1970.
[90] Mayr, E. W., Meyer, A. R.: The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals. M.I.T.: Lab. Comput. Sci. Rep. LCS/TM-199 (1981).
[91] McDonald, I. D.: A Computer Application to Finite p-Groups. J. Aust. Math. Soc. **17**, 102–112 (1974).
[92] Möller, H. M.: Mehrdimensionale Hermite-Interpolation und numerische Integration. Math. Z. **148**, 107–118 (1976).

[93] Moses, J.: Symbolic Integration. Ph.D. Thesis, Cambridge, Mass., M.I.T., Math. Dept., Rep. MAC-**47**, 1967.
[94] Moses, J.: Algebraic Simplification: A Guide for the Perplexed. SYMSAM **1971**, 282–304 and Commun. ACM **14**, 527–537 (1971).
[95] Musser, D. R.: Algorithms for Polynomial Factorization. Univ. of Wisconsin, Madison: Ph.D. Thesis, Techn. Rep. **134**, 1971.
[96] Musser, D. R.: A Data Type Verification System Based on Rewrite Rules. USC Information Science Institute, Marina del Rey, Calif.: 1977.
[97] Musser, D. R.: Convergent Sets of Rewrite Rules for Abstract Data Types. Information Sciences Institute: Report 1978.
[98] Musser, D. R.: On Proving Inductive Properties of Abstract Data Types. Seventh ACM Symp. on Principles of Programming Languages **1980**, 154–162.
[99] Newman, M. F.: Calculating Presentations for Certain Kinds of Quotient Groups. SYMSAC **1976**, 2–8.
[100] Newman, M. H. A.: On Theories with a Combinatorial Definition of "Equivalence". Ann. Math. **43**/2, 223–243 (1942).
[101] O'Donnell, M.: Computing in Systems Described by Equations. Lecture Notes in Computer Science, Vol. 58. Berlin-Heidelberg-New York: Springer 1977.
[102] Oldehoeft, A.: Analysis of Constructed Mathematical Responses by Numeric Tests for Equivalence. Proc. ACM 24th Nat. Conf., 117–124 (1969).
[103] Paterson, M. S., Wegmann, M. N.: Linear Unification. J. Comput. Syst. Sci. **16**, 158–167 (1978).
[104] Pavelle, R., Rothstein, M., Fitch, J.: Computer Algebra. Preprint, Scientific American 1981.
[105] Peterson, G. E., Stickel, M. E.: Complete Set of Reductions for Some Equational Theories. J. ACM **28**/2, 233–264 (1981).
[106] Plaisted, D.: Well-Founded Orderings for Proving Termination of Systems of Rewrite Rules. Univ. of Illinois, Urbana-Champaign: Rep. 78-**932**, 1978.
[107] Plaisted, D.: A Recursively Defined Ordering for Proving Termination of Term Rewriting Systems. Univ. of Illinois, Urbana-Champaign: Rep. 78-**943**, 1978.
[108] Plotkin, G.: Building-In Equational Theories. Machine-Intelligence **7**, 73–90 (1972).
[109] Pohst, M., Yun, D. Y. Y.: On Solving Systems of Algebraic Equations Via Ideal Bases and Elimination Theory. SYMSAC **1981**, 206–211.
[110] Polak, W.: Program Verification at Stanford: Past, Present, Future. Stanford University: Comput. Syst. Lab., Rep. 1981.
[111] Raulefs, P., Siekmann, J., Szabó, P., Unvericht, E.: A Short Survey on the State of the Art in Matching and Unification Problems. SIGSAM Bull. **13**/2, 14–20 (1979).
[112] Reynolds, J. C.: Reasoning About Arrays. Commun. ACM **22**/5, 290–299 (1979).
[113] Richardson, D.: Some Unsolvable Problems Involving Elementary Functions of a Real Variable. J. Symb. Logic **33**, 511–520 (1968).
[114] Richardson, D.: Solution of the Identity Problem for Integral Exponential Functions. Math. Logik Grundlagen Math. **15**, 333–340 (1969).
[115] Richter, M. M., Kemmenich, S.: Reduktionssysteme und Entscheidungsverfahren. Rhein.-Westf. Tech. Hochschule, Aachen: Inst. f. Informatik, Rep. **4**, 1980.
[116] Risch, R. H.: The Problem of Integration in Finite Terms. Trans. AMS **139**, 167–189 (1969).
[117] Robinson, J. A.: A Machine-Oriented Logic Based on the Resolution Principle. J. ACM **12**, 23–41 (1965).
[118] Robinson, J. A.: Computational Logic: The Unification Computation. Machine Intelligence **6**, 63–72. New York: Edinb. Univ. Press 1971.
[119] Rogers, H.: Theory of Recursive Functions and Effective Computability. New York: McGraw-Hill 1967.
[120] Rosen, B. K.: Tree-Manipulating Systems and Church-Rosser Theorems. J. ACM **20**/1, 160–187 (1973).
[121] Schaller, S.: Algorithmic Aspects of Polynomial Residue Class Rings. Ph.D. Thesis, Comput. Sci. Tech., University of Wisconsin, Madison, Rep. **370**, 1979.
[122] Schrader, R.: Zur konstruktiven Idealtheorie. Dipl. Thesis, Math. Inst., Univ. of Karlsruhe, Federal Republic of Germany 1976.
[123] Seidenberg, A.: Constructions in Algebra. Trans. AMS **197**, 273–313 (1974).

[124] Sethi, R.: Testing for the Church-Rosser Property. J. ACM **21**/4, 671–679 (1974), **22**/3, 424 (1975).
[125] Shtokhamer, R.: Simple Ideal Theory: Some Applications to Algebraic Simplification. Univ. of Utah, Salt Lake City: Tech. Rep. UCP-**36**, 1975.
[126] Shtokhamer, R.: A Canonical Form of Polynomials in the Presence of Side Relations. Technion, Haifa: Phys. Dept., Rep. **25**, 1976.
[127] Shtokhamer, R.: Attempts in Local Simplification of Non-Nested Radicals. SIGSAM Bull. **11**/1, 20–21 (1977).
[128] Spear, D.: A Constructive Approach to Commutative Ring Theory. MACSYMA **1977**, 369–376.
[129] Staples, J.: Church-Rosser Theorems for Replacement Systems. Algebra and Logic (Crossley, J., ed.), Lecture Notes in Mathematics, Vol. 450, pp. 291–307. Berlin-Heidelberg-New York: Springer 1975.
[130] Stickel, M. E., Peterson, G. E.: Complete Sets of Reductions for Equational Theories with Complete Unification Algorithm. Univ. of Arizona: Dept. Comp. Sci. 1977.
[131] Stoutemyer, D. R.: Qualitative Analysis of Mathematical Expressions Using Computer Symbolic Mathematics. SYMSAC **1976**, 97–104.
[132] Stoutemyer, D. R.: Automatic Simplification for the Absolute Value Function and its Relatives. ACM SIGSAM Bull. **10**/4, 48–49 (1976).
[133] Suzuki, N., Jefferson, D.: Verification Decidability of Presburger Array Programs. J. ACM **27**/1, 191–205 (1980).
[134] Szekeres, G.: A Canonical Basis for the Ideals of a Polynomial Domain. Am. Math. Mon. **59**/6, 379–386 (1952).
[135] Tobey, R. G.: Experience with FORMAC Algorithm Design. Commun. ACM **9**, 589–597 (1966).
[136] Todd, J. A., Coxeter, H. S. M.: A Practical Method for Enumerating Cosets of a Finite Abstract Group. Proc. Edinb. Math. Soc. (2) **5**, 26–34 (1936).
[137] Trinks, W.: Über B. Buchbergers Verfahren, Systeme algebraischer Gleichungen zu lösen. J. Number Theory **10**/4, 475–488 (1978).
[138] Trotter, P. G.: Ideals in $Z[x,y]$. Acta Math. Acad. Sci. Hungar. **32** (1–2), 63–73 (1978).
[139] Winkler, F., Buchberger, B., Lichtenberger, F., Rolletschek, H.: An Algorithm for Constructing Canonical Bases (Gröbner-Bases) for Polynomial Ideals. (Submitted.)
[140] Yun, D. Y. Y., Stoutemyer, R. D.: Symbolic Mathematical Computation. Encyclopedia of Computer Science and Technology (Belzer, J., Holzman, A. G., Kent, A., eds.), Vol. 15, pp. 235–310. New York-Basel: Marcel Dekker 1980.
[141] Zacharias, G.: Generalized Gröbner Bases in Commutative Polynomial Rings. Bachelor Thesis, Dept. Comput. Sci., M.I.T., 1978.
[142] Zippel, R. E. B.: Simplification of Radicals with Applications to Solving Polynomial Equations. Cambridge, Mass.: M.I.T., Dept. of Electrical Engineering and Computer Science, Master's Thesis, 1977.

Prof. Dr. B. Buchberger
Institut für Mathematik
Johannes-Kepler-Universität Linz
Altenbergerstrasse 69
A-4040 Linz
Austria

Prof. Dr. R. Loos
Institut für Informatik I
Universität Karlsruhe
Zirkel 2
D-7500 Karlsruhe
Federal Republic of Germany