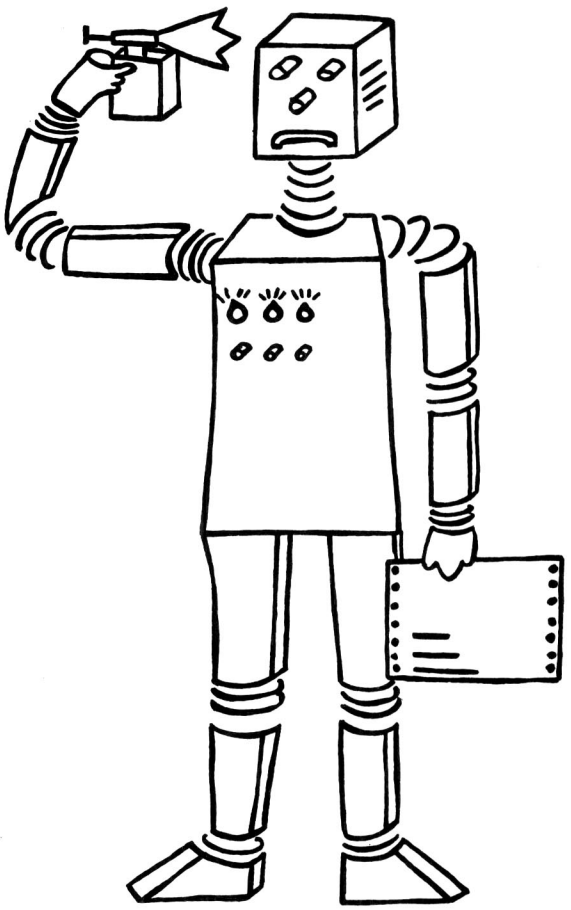
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany

SEKI-Working Paper



A Decision Procedure for Presburger Arithmetic with Functions and Equality

F.-J. Krämer

SEKI Working Paper SWP-89-4

A Decision Procedure for Presburger Arithmetic with Functions and Equality

F.-J. Krämer
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, D-6750 Kaiserslautern, W.-Germany

Table of contents

Preface

Many of the formulas one tends to encounter in program verification are contained in the quantifier-free Presburger Arithmetic and its extension by predicate and function symbols.
Remarkable work for both the unextended and the extended class has been done by Robert Shostak. Particularly for the quantifier-free Presburger Arithmetic, he developed a very elegant algorithm based on the computation of loop residues in a graph. The algorithm decides the satisfiability of of a conjunction of atomic formulas of this theory.
This diploma thesis presents the development of a decision procedure for the satisfiability of a conjunction of atomic formulas of the quantifier-free Presburger Arithmetic extended by predicate and function symbols:
The EQUALITY_LOOP_RESIDUE procedure.
This procedure combines both the concept of the LOOP_RESIDUE method in a refined version adjusted to the specific problems of the extended class and the concept of rewriting.
The connection of this decision problem to ground terms is pointed out and a polynomial time algorithm for computing a canonical rewriting system equivalent to a finite set of ground equations recently (1988) presented by Gallier, Narendran, Plaisted, Raatz, and Snyder can be used as subprocedure in the newly developed decision procedure.
The EQUALITY_LOOP_RESIDUE procedure does not only improve the decision process of the satisfiability of a conjunction of atomic formulas, but also the determination of the validity of a quantifier-free Presburger formula including predicate and function symbols.

1. Introduction

This diploma thesis deals with the problem of deciding the validity of a formula out of a certain language class and of an extension of this class.
It is the language of quantifier-free Presburger Arithmetic: it contains all Presburger formulas without quantifiers. The extended class[1] includes in addition formulas with (uninterpreted) predicate and function symbols. The interpreted predicate and function symbols of the Presburger Arithmetic are $<$, and 0, 1, and $+$, respectively.

Array bound checks and tests on index variables, for example, give rise to formulas of $\mathcal{PA}$ during program verification. And programs that operate on arrays and other data structures that can be modeled as uninterpreted functions are prone to produce formulas of $\mathcal{PA}_f$ during the verification process of the program.

Therefore, many formulas occurring during program verification and in theorem provers belong to the unextended class $\mathcal{PA}$ and even more belong to the extended class $\mathcal{PA}_f$. And since the decision of such formulas constitutes the main issue in such programs, this straightforwardly indicates the motivation for the development of efficient decision procedures.

It is evident by negation and expansion into disjunctive normal form that it suffices to provide a decision procedure for the satisfiability of a conjunction of atomic formulas in order to decide the validity of a quantifier-free formula.

Robert Shostak [27] has observed that the formulas occurring in program verification have in many cases a simple form, and has developed an elegant method for deciding the satisfiability of a (quantifier-free) conjunction of atomic Presburger formulas where each formula can be written in the form $ax + by \leq c$ in which a, b, and c are constants, and x, and y are variables.

The underlying data structure of this method is a graph which is constructed in such a way that each variable x labels a vertex and each atomic formula $ax + by \leq c$ labels an edge incidenting with those vertices labeled by x and y.
The LOOP_RESIDUE method has basically two concepts:
The computation of loop residues and the notion of a closure of a graph.

1: For convenience and abbreviation, we refer to the quantifier-free Presburger Arithmetic by $\mathcal{PA}$ or unextended class and to its extension including predicate and function symbols by $\mathcal{PA}_f$ or extended class.

For example, a loop labeled by $x - y \leq 0$, $y - z \leq 0$, and $z - x \leq c$ has residue $x - x \leq c$. The atomic formula $x - x \leq c$ is unsatisfiable for $c<0$. The notion of a closure of such a graph is essential for the completeness of the method.
The treatment of atomic formulas with relation $<$ and of atomic formulas with an arbitrary number of variables has for simplicity in presenting the method been postponed as generalization. The generalized procedure can determine the satisfiability of any conjunction of atomic formulas and therefore provides a decision procedure for the quantifier-free Presburger Arithmetic. It combines generality as well as suitability for simply structured formulas and is described in the second chapter.

Another method for determining the satisfiability of a (quantifier-free) conjunction of atomic formulas is the SUP-INF method. Shostak attributes the method to Bledsoe [3] and presents a refined version in [28].
At the core of the method are two procedures SUP and INF which compute the maximum and minimum real value that a variable can have subject to a set of linear constraints.
Shostak's basic version of the SUP-INF method is complete for the real domain, and incomplete for the integer domain. But, "the incompleteness of the basic method is of so little practical consequence, however, that schemes for augmenting the method in order to obtain completeness are largely a matter of theoretical interest." [28]
The method can easily be applied to linear maximization problems with a linear objective function M and a set L of linear constraints by adding $z \leq M$ as a constraint where z is a new variable and computing the maximal (real) value of z subject to $L \cup \{z \leq M\}$ with the procedure SUP. Similiarly, if M is to be minimized, $M \leq z$ is added and the procedure INF determines the minimal real value of z subject to the set $L \cup \{M \leq z\}$ of constraints.
In order to compare the SUP-INF method with well-known real and integer linear programming algorithms, Shostak implemented "both the Simplex algorithm for real linear programming and the Gomory cutting plane algorithm for integer linear programming. Surprisingly, neither of these implementations performed more efficiently than the SUP-INF implementation on naturally arising formulas. The rather poor relative performance of the well-known algorithms stems from the fact that typical problems tend to be small (thus emphasizing the overhead necessary to convert the problem to matrix form) and structured in such a way as to produce sparse matrices. The Gomory algorithm, moreover, could not be trusted to terminate on unbounded problems." [28]

Related to the decision problem of the extended class is the decision problem of the quantifier-free theory of equality. The atomic formulas of this language are equalities of the form $t=t'$ where t and t' are terms.
The problem of verifying that an equality is a consequence of several other equalities is a decision problem of the quantifier-free theory of equality. For example, $f(f(a,b),b)=a$ is a consequence of $f(a,b)=a$, or, less obviously, $fa=a$ is a consequence of $fffa=a$ and $fffffa=a$.
Again, since we have no quantifiers, it suffices to provide a decision procedure for the satisfiability of a conjunction of atomic formulas in order to decide the validity of formulas of this theory.
"A practical algorithm for this problem is essential to mechanical program verification (or to any other kind of mechanical reasoning), since almost all proofs require reasoning about equalities." [17]
The decision problem for the quantifier-free theory of equality with uninterpreted function symbols has been attacked from quite different points of view: As a variation of the common subexpression problem by Downey, Sethi, and Tarjan [6], and as the word problem in finitely presented algebras by Kozen [13].
Nelson and Oppen reduce all those problems "to the problem of constructing the 'congruence closure' of a relation on a graph." [17]
The construction of the congruence closure of a relation on a graph is described in the third chapter.

An approach for the extended class for the integer domain was given by Shostak in [25] and is presented in the forth chapter. It introduces to the main problem that arises from the extension by function-symbols: The well-definedness of functions: A function is well defined if it assigns to every argument of its domain a uniquely defined value of its range.[2]

Given a conjunction C of atomic formulas, Shostak obtains an associated integer linear programming problem (ILP) by replacing the terms occurring in the conjunction by new variables. An ILP-solver then determines the satisfiability of the associated problem. Obviously, if the associated problem has no solution, then the given conjunction C is unsatisfiable.
Otherwise, the ILP-solver constructs a solution for the associated problem and another procedure then examines this solution for violations of well definedness, or, in Shostak's words, for violations of the substitutivity of equality subject to the conjunction C. An appropriate formula that

2: Shostak denotes the well definedness as the property of substitutivity of equality: if two terms are equal, they can replace each other mutually as arguments of a function without changing its value.

summarizes the violation is generated, added to the conjunction C, and the process is reapplied for each conjunction of the now resulting disjunctive normal form.
The proof of the decidability of the extended class in chapter four provides itself a decision procedure, but Shostak's suggested procedure reduces the combinatorial explosion substantially. Nevertheless, there is still enough opportunity for further improvement.

The objective of this diploma thesis is the design of a method for the extended class which is similiarly suitable to simply structured formulas like the LOOP_RESIDUE method for the unextended class. This procedure should play a central part in the extended procedure.
As noted above, the main problem here is the well definedness of function symbols. We basically succeeded in the design of such an algorithm by means both of compactness in representing information and of incorporation of other concepts, especially rewriting:
A set $\mathcal{E}$ of equalities which are implied by a conjunction C of atomic formulas is computed such that any equality implied by C is implied by $\mathcal{E}$. A canonical rewriting system $\mathcal{R}$ equivalent to $\mathcal{E}$ is computed and the conjunction C is reduced wrt the system $\mathcal{R}$. The LOOP_RESIDUE method can now immediately be applied to the resulting conjunction.
Briefly spoken, this is the basic idea of the presented procedure which is named EQUALITY_LOOP_RESIDUE procedure.
And this procedure seems to improve the decision process not only for large, but also for small problems. Violations of well definedness are prevented implicitely while Shostak has done this explicitely.

1.1. Language classes: a survey

Figure 1.1 shows the location of the quantifier-free Presburger Arithmetic and its extension in the hierarchy of language classes.
The language of second order predicate logic is neither recursively decidable nor recursively enumerable. The one of first order is also not recursively decidable, but recursively enumerable.
A subset of this language is the Presburger Arithmetic which has the interpreted function symbols 0, 1, and + and the interpreted predicate symbol $<$.
This class was first shown to be decidable by Presburger in 1929 [22].
A well known decision procedure, described by Kreisel and Krevine [14], is based on a method of quantifier-elimination and prone to combinatorial explosion. A more efficient procedure has been given by Cooper [4] in

1972. It is probably the best one in the worst case, and has deterministic time complexity $O(2^{2^{2^n}})$ where n is the length of the formula. This was shown by Oppen [19] in 1975.

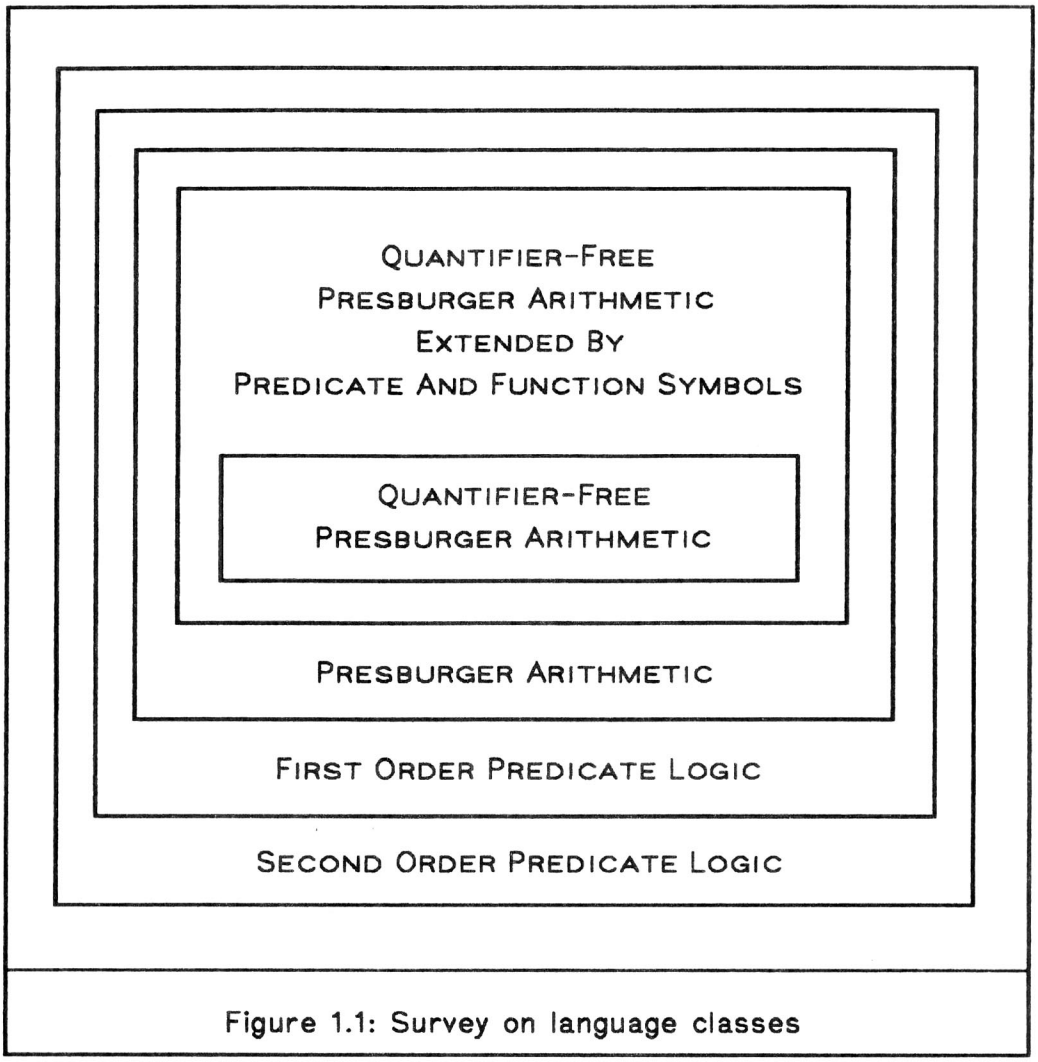


Figure 1.1: Survey on language classes

Those Presburger formulas without quantifiers define the quantifier-free Presburger Arithmetic. As a subclass of the Presburger Arithmetic, it surely is decidable. This is also true for the extension by predicate and function symbols: Any formula of the extended language can be transformed into an equivalent formula of the unextended language. For both language classes $\mathcal{PA}$ and $\mathcal{PA}_f$, the decision problem can deterministically be solved in worst case exponential time.

Please note that the Presburger Arithmetic does not include the extension of the quantifier-free Presburger Arithmetic because it does not include formulas with function symbols. But it includes for every such formula an equivalent formula.

1.2. Motivation

Consider that you want to write a program and prove its correctness wrt certain input and output specifications. Briefly spoken, one way to achieve this is to state a precondition for the input data and a postcondition which connects the input data with the output data and finally to prove the validity of an appropriate formula which includes those conditions and the semantics of the program.[3]
So the task of program verification splits up into generating the corresponding formula and checking its validity.
For example, consider the PASCAL-like program that transforms a bit string into a decimal number which is shown in figure 1.2.

```
{ PRECONDITION:  n ∈ ℕ;  a ∈ (ℕ → ℕ) }
        D := 0
        i := n + 1
        WHILE 0 < i DO
        BEGIN
            D := 2*D + a(i-1)
            i := i - 1
        END
{ POSTCONDITION:  D = Σ_{j=0}^{n} 2^j * a(j) }
```

Figure 1.2.: Program with Pre- and Postcondition

In order to prove the correctness of the program in figure 1.2. wrt the pre- and postcondition, it is necessary to construct an appropriate formula and to prove its validity. As a part of this process, it is important to find a (WHILE) loop invariant formula. Loop invariant formula means that if it holds before entrance into the loop then it holds after exit of the loop.

3: The pioneering paper is written by Hoare [10] titled "An axiomatic basis for computer programming". Detailed information is found in "Foundations of Program Verification" presented by Loeckx and Sieber [15].

For example, define a formula $\mathcal{F}$ by

$$\mathcal{F} \equiv \mathcal{F}(a, i, n, D) \equiv (D = \sum_{\mathcal{J}=i}^{n} 2^{\mathcal{J}-i} * a(\mathcal{J}) \wedge i \in \mathbb{N}).$$

Since the formula $\mathcal{F}_{\mathcal{D}}$ given by

$$\mathcal{F}_{\mathcal{D}} \equiv (\mathcal{F}(a, i, n, D) \wedge 0 < i \rightarrow \mathcal{F}(a, i-1, n, 2*D + a(i-1))).$$

or, equivalently, by replacing the occurences of $\mathcal{F}$ in $\mathcal{F}_{\mathcal{D}}$,

$$\mathcal{F}_{\mathcal{D}} \equiv [((D = \sum_{\mathcal{J}=i}^{n} 2^{\mathcal{J}-i} * a(\mathcal{J}) \wedge i \in \mathbb{N}) \wedge 0 < i)$$
$$\rightarrow (2*D + a(i-1) = \sum_{\mathcal{J}=i-1}^{n} 2^{\mathcal{J}-(i-1)} * a(\mathcal{J}) \wedge (i-1) \in \mathbb{N})]$$

is valid, the formula $\mathcal{F}$ is an invariant formula of the WHILE loop.
Observe that this formula contains, in addition to the function symbols 0, 1, and + of the Presburger theory, the function symbol a.
Since multiplication by constants is used in abbreviation for repeated addition, the formula $\mathcal{F}$ is contained in the language class $\mathcal{PA}_f$ of quantifier-free Presburger formulas extended by predicate and function symbols.
An array is one example of data structures that can be modeled by uninterpreted function symbols. All such data structures that can be modeled by uninterpreted function symbols give rise to formulas of the quantifier-free Presburger Arithmetic extended by predicate and function symbols in the process of program verification.

2. Quantifier-free Presburger Arithmetic

As a subclass of the Presburger Arithmetic, the quantifier-free Presburger Arithmetic surely is decidable. It's decision complexity is no worse than exponential and is therefore substantially easier to decide than full Presburger Arithmetic, which is decidable in $O(2^{2^{2^n}})$.
A commonly used technique to prove the validity of a quantifier-free formula $\mathcal{F}$ is to prove the unsatisfiability of each (quantifier-free) conjunction in the disjunctive normal form of the negation of $\mathcal{F}$ where each such conjunction consists of atomic formulas.
The atomic formulas of the quantifier-free Presburger Arithmetic can be written in the form $a_1v_1+a_2v_2+\ldots+a_nv_n\ rel\ a_0$ where $rel \in \{\leq, <, =, \neq\}$, the v_i's are variables, and the a_i's are constants.
A decision procedure for the satisfiability of conjunctions of linear lequalities[4] with at most two variables ($ax+by\leq c$) is presented in section 2.2 and is generalized to decide the satisfiability of terms of the form $a_1v_1+a_2v_2+\ldots+a_nv_n\ rel\ a_0$ where $rel \in \{\leq, <, =, \neq\}$.

2.1. The Theory

We first give a definition of the Presburger Arithmetic according to Harrison [9].
The set $\mathcal{PA}$ of Presburger Formulas wrt the domain $\mathcal{D}$ is the least class satisfying the following conditions:

a) For every $m\geq 0$ and $n_i, n_i' \in \mathcal{D}$, $0\leq i\leq m$,

$$n_0 + n_1x_1 + \ldots + n_mx_m = n_0' + n_1'x_1 + \ldots + n_m'x_m$$

is a formula in $\mathcal{PA}$ with free variables $x_1, \ldots, x_m$.
b) If $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{PA}$, then so is $\mathcal{F}_1 \wedge \mathcal{F}_2$.
c) If $\mathcal{F}_1, \mathcal{F}_2 \in \mathcal{PA}$, then so is $\mathcal{F}_1 \vee \mathcal{F}_2$.
d) If $\mathcal{F} \in \mathcal{PA}$, then so is $\neg\mathcal{F}$.
e) If $\mathcal{F}(x_1,\ldots,x_n) \in \mathcal{PA}$ and $1\leq i\leq n$, then $(\forall x_i)\mathcal{F}(x_1,\ldots,x_n) \in \mathcal{PA}$.
f) If $\mathcal{F}(x_1,\ldots,x_n) \in \mathcal{PA}$ and $1\leq i\leq n$, then $(\exists x_i)\mathcal{F}(x_1,\ldots,x_n) \in \mathcal{PA}$.

A formula with no free variable is called a *sentence*.
Presburger defined the Presburger Formulas wrt the domain of the non-negative integers, the natural numbers $\mathbb{N}$, and first proved the decidability of this class in 1929 [22]. But instead of using a multiplication symbol, he replaced the multiplication with constant natural numbers by repeated

4: Since equality and inequality refer to the relations = and ≠, respectively, we refer to ≤ by lequality (less or equal).

addition. A later proof of the decidability of the Presburger theory can be found in Yasuhara's monograph of 1971 [32].
Formulas of the Presburger Arithmetic like $x+x+x+x+x$ are conveniently abbreviated by $5*x$ or $5x$, for example, though the multiplication symbol $*$ is not contained in the original definition of the Presburger Arithmetic.

Not only for the initial domain of the Presburger theory, the quantifier-free Presburger Theory (conditions a), b), c), and d)) is also decidable wrt the domain of the integer numbers $\mathbb{Z}$, the rational numbers $\mathbb{Q}$, and the real numbers $\mathbb{R}$.[5]

And for the domain of the rational numbers, a formula $\mathcal{F}$ can equivalently be transferred into one where the coefficients are all integers. This is simply done by multiplying each atomic formula occurring in $\mathcal{F}$ with the main denominator of the rational constants occurring in the atomic formula. This indicates that the multiplication symbol can also be replaced for this domain.

The natural, integer, rational, and real numbers are all infinite linearly ordered domains. The notion of a *linearly ordered domain* is given in [32] by the following conditions for a predicate $<$:

$$(\forall x)(\forall y)\ (x<y \vee x=y \vee y<x)$$
$$(\forall x)(\forall y)(\forall z)\ (x<y \wedge y<z \supset x<z)$$
$$(\forall x)\neg(x<x)$$

A domain is *dense* if the following condition is satisfied:

$$(\forall x)(\forall z)(\exists y)\ (x<z \supset x<y \wedge y<z)$$

The rational and real numbers are dense, while the natural and integer numbers are not dense. The LOOP_RESIDUE procedure presented in the following section is complete only for dense domains like $\mathbb{Q}$ or $\mathbb{R}$. For domains which are not dense like the integer numbers, the procedure may return satisfiable for an unsatisfiable set wrt this domain. Therefore, the density of the domain is a necessary condition for the completeness.

2.2. The Loop-Residue Method

2.2.1. Requirements

The springs for the development of another method, generality and simplicity, are neatly expressed in the following quotation:
"A number of approaches have been used to decide the feasibility of sets of inequalities, ranging from goal-driven rewriting mechanisms to the power-

5: Shostak gives decision procedures for these problems in [27] and [25].

ful simplex techniques of linear programming. Some simple methods are well suited to the small, trivial problems that most often arise, but are insufficiently general. Full-scale simplex techniques, on the other hand, are general and fast for medium to large scale problems, but do not take advantage of the trivial structure of the small problems (involving only a few variables and equations) encountered most frequently in program verification and related applications." [27] [6]

2.2.2. The main theorem for the unextended class

Denote by $\mathcal{L}$, or $\mathcal{L}_{\leq}$, a set of linear lequalities[7] each of whose elements l, or $l_{\leq}$, can be written in the form $ax+by \leq c$, where x and y are variables and a, b, and c are constants.
Obtain $\mathcal{L}_{<}$, $\mathcal{L}_{=}$, $\mathcal{L}_{>}$, $\mathcal{L}_{\geq}$, and $\mathcal{L}_{\neq}$ from $\mathcal{L}$ and $l_{<}$, $l_{=}$, $l_{>}$, $l_{\geq}$, and $l_{\neq}$ from l by replacing $\leq$ by $<$, $=$, $>$, $\geq$, and $\neq$, respectively.
We choose as domain $\mathcal{D}$ the dense linearly ordered structure of the rational numbers $\mathbb{Q}$.[8]
For convenience, a special variable v_0 is introduced as zero variable: it is assumed that it appears only with coefficient zero, while the other variables require - without loss of generality - nonzero coefficients.

Define $G(\mathcal{L})$ as the graph for a set $\mathcal{L}$ of linear lequalities in the following way:

For each variable occurring in $\mathcal{L}$ give $G(\mathcal{L})$ a vertex labeled with this variable and for each linear lequality in $\mathcal{L}$ give $G(\mathcal{L})$ an edge labeled with this lequality such that this edge connects the corresponding vertices of the occurring variables.

Denote, in addition, for a graph G whose edges are labeled by lequalities the set of those lequalities by $\mathcal{L}_G$, or $\mathcal{L}_{G\leq}$.
A *path* $\mathcal{P}$ through G is uniquely described by a sequence $v_1, v_2, \ldots v_n, v_{n+1}$ of vertices and a sequence $e_1, e_2, \ldots, e_n$ of edges ($n \geq 1$).

6: Atomic formulas of the form $ax+by \leq c$ are denoted as lequalities in this diploma thesis in order to distinguish from inequalities $ax+by \neq c$. But Shostak denotes atomic formulas of the form $ax+by \leq c$ also as inequalities.

7: The number of variables per inequality is restriced to two for convenience. As an extension of the procedure, this restriction is dropped.

8: It will become evident that the density of the domain is necessary for the completeness of the procedure. The Loop-Residue procedure is thus incomplete for the integers.

Denote by the *triple sequence* for $\mathcal{P}$ the sequence $\langle a_1,b_1,c_1\rangle$, $\langle a_2,b_2,c_2\rangle$, ... $\langle a_n,b_n,c_n\rangle$ where $a_iv_i+b_iv_{i+1}\leq c_i$ is the lequality l_i labeling edge e_i for each i, $1\leq i\leq n$, and by $\mathcal{L}_{\mathcal{P}}$, or $\mathcal{L}_{\mathcal{P}\leq}$, the set of linear lequalites labeling $\mathcal{P}$.
A path $\mathcal{P}$ is *admissible* if for all $1\leq i\leq n-1$, b_i and a_{i+1} of its triple sequence have opposite signs, i.e., $sgn(b_i*a_{i+1})=-1$.[9]

Define the *binary operator* $*$ on triples as follows:
For two triples $\langle a_1,b_1,c_1\rangle$ and $\langle a_2,b_2,c_2\rangle$ where b_1 and a_2 have opposite signs, define $\langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle := \langle ka_1a_2, -kb_1b_2, k(c_1a_2-c_2b_1)\rangle$ (or, in abbreviation, $(k\langle a_1a_2, -b_1b_2, c_1a_2-c_2b_1\rangle)$) where $k=sgn(a_2)$.[9]

Lemma 2.1.
The binary operator $*$ on triples is associative.
Proof.

$$
\begin{aligned}
&(\langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle) * \langle a_3,b_3,c_3\rangle \\
&= \langle k_2a_1a_2, -k_2b_1b_2, k_2(c_1a_2-c_2b_1)\rangle * \langle a_3,b_3,c_3\rangle \\
&= \langle k_3(k_2a_1a_2)a_3, -k_3(-k_2b_1b_2)b_3, k_3(k_2(c_1a_2-c_2b_1)a_3 - c_3(-k_2b_1b_2)\rangle \\
&= \langle k_2a_1(k_3a_2a_3), -k_2b_1(-k_3b_2b_3), k_2(c_1(k_3a_2a_3) - k_3(c_2a_3-c_3b_2)b_1)\rangle \\
&= \langle a_1,b_1,c_1\rangle * \langle k_3a_2a_3, -k_3b_2b_3, k_3(c_2a_3-c_3b_2)\rangle \\
&= \langle a_1,b_1,c_1\rangle * (\langle a_2,b_2,c_2\rangle * \langle a_3,b_3,c_3\rangle)
\end{aligned}
$$

where $k_2=sgn(a_2)$ and $k_3=sgn(a_3)$. □

The *residue* $r_{\mathcal{P}} = \langle a_{\mathcal{P}}, b_{\mathcal{P}}, c_{\mathcal{P}}\rangle$, or $r(\mathcal{P}) = \langle a(\mathcal{P}), b(\mathcal{P}), c(\mathcal{P})\rangle$, of an admissible path $\mathcal{P}$ with triple sequence $\langle a_1,b_1,c_1\rangle$, $\langle a_2,b_2,c_2\rangle$, ... $\langle a_n,b_n,c_n\rangle$ is defined by

$$\langle a_{\mathcal{P}}, b_{\mathcal{P}}, c_{\mathcal{P}}\rangle := \langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle * \ldots * \langle a_n,b_n,c_n\rangle .$$

The residue of an admissible path is uniquely defined since $*$ is associative.

The *residue lequality* $l_{\mathcal{P}}$, $l(\mathcal{P})$, or $l_{\mathcal{P}\leq}$, of an admissible path $\mathcal{P}$ is given by $a(\mathcal{P})x + b(\mathcal{P})y \leq c(\mathcal{P})$, where $\langle a(\mathcal{P}), b(\mathcal{P}), c(\mathcal{P})\rangle$ is the residue of $\mathcal{P}$ and x and y are the first and last vertices, respectively, of $\mathcal{P}$.

Lemma 2.2.
For an admissible path $\mathcal{P}$,
i) $\mathcal{L}_{\mathcal{P}}$ and $\mathcal{L}_{\mathcal{P}} \cup \{l_{\mathcal{P}}\}$ are equivalid,
ii) $\mathcal{L}_{\mathcal{P}<}$ and $\mathcal{L}_{\mathcal{P}<} \cup \{l_{\mathcal{P}<}\}$ are equivalid, and
iii) $\mathcal{L}_{\mathcal{P}=}$ and $\mathcal{L}_{\mathcal{P}=} \cup \{l_{\mathcal{P}=}\}$ are equivalid.

9: The function sgn returns the sign of a number and is defined by $sgn(0)=0$, $sgn(x)=1$ for $x>0$, and $sgn(x)=-1$ for $x<0$.

Proof.

i) This is done by induction on the length n of path $\mathcal{P}$.

Basis: $n=2$.

Let x, y, and z be the first, second, and last vertex of $\mathcal{P}$, respectively.

$\mathcal{L}_{\mathcal{P}} = \{ a_1x + b_1y \le c_1,\ a_2y + b_2z \le c_2 \}$ is valid

iff (observe that $b_1 \neq 0$ and $a_2 \neq 0$)

$\{ sgn(a_2)a_1a_2x + sgn(a_2)b_1a_2y \le sgn(a_2)c_1a_2,$
$sgn(b_1)b_1a_2y + sgn(b_1)b_1b_2z \le sgn(b_1)b_1c_2 \}$ is valid

iff (since by admissibility of $\mathcal{P}$, $sgn(b_1a_2)=-1$)

$\{ sgn(a_2)a_1a_2x + sgn(a_2)b_1a_2y \le sgn(a_2)c_1a_2,$
$-sgn(a_2)b_1a_2y - sgn(a_2)b_1b_2z \le -sgn(a_2)b_1c_2 \}$ is valid

iff

$\mathcal{L}_{\mathcal{P}} \cup \{l_{\mathcal{P}}\}$ is valid.

where $l_{\mathcal{P}}$ is the residue lequality of $\mathcal{P}$, $a(\mathcal{P})x + b(\mathcal{P})z \le c(\mathcal{P})$, or, equivalently, $sgn(a_2)a_1a_2x - sgn(a_2)b_1b_2z \le sgn(a_2)(c_1a_2 - c_2b_1)$.

Induction Step. $n>2$. Let $\mathcal{Q}$ and $\mathcal{R}$ paths such that $\mathcal{P}=\mathcal{Q}\mathcal{R}$. Then,

$\mathcal{L}_{\mathcal{P}} = \mathcal{L}_{\mathcal{Q}} \cup \mathcal{L}_{\mathcal{R}}$ is valid

iff (by the induction hypothesis for $\mathcal{Q}$ and $\mathcal{R}$)

$\mathcal{L}_{\mathcal{Q}} \cup \{l_{\mathcal{Q}}\} \cup \mathcal{L}_{\mathcal{R}} \cup \{l_{\mathcal{R}}\}$ is valid

iff (since $\{l_{\mathcal{Q}}, l_{\mathcal{R}}\}$ is valid if and only if $\{l_{\mathcal{Q}}, l_{\mathcal{R}}, l_{\mathcal{P}}\}$ is valid by the same arguments as in the basis)

$\mathcal{L}_{\mathcal{Q}} \cup \mathcal{L}_{\mathcal{R}} \cup \{l_{\mathcal{Q}}, l_{\mathcal{R}}, l_{\mathcal{P}}\}$ is valid

iff

$\mathcal{L}_{\mathcal{P}} \cup \{l_{\mathcal{P}}\}$ is valid

completing the proof of i).

ii, iii) The same proof as for i) with $<$ and $=$ instead of the relation $\le$. □

A path with identical first and last vertex is called a *loop*. A loop is *simple* if its intermediate vertices are distinct.

An admissible loop $\mathcal{P}$ is called an *equality loop* if and only if $a_{\mathcal{P}}+b_{\mathcal{P}}=0$ and $c_{\mathcal{P}}=0$ and an *infeasible loop* if and only if $a_{\mathcal{P}}+b_{\mathcal{P}}=0$ and $c_{\mathcal{P}}<0$, where $\langle a_{\mathcal{P}}, b_{\mathcal{P}}, c_{\mathcal{P}} \rangle$ is the loop residue of $\mathcal{P}$.

The residue lequality $a(\mathcal{P})x + b(\mathcal{P})x \le c(\mathcal{P})$ of an infeasible (simple) loop with initial and final vertex x is unsatisfiable. It follows that the set $\mathcal{L}$ of linear lequalities is unsatisfiable if the graph $G(\mathcal{L})$ has an infeasible simple loop.

The converse does not hold as the following example demonstrates:
Consider the set $\mathcal{L} = \{x \geq 2,\ x \leq y,\ x+y \leq 2\}$ which is unsatisfiable. The only admissible loop in $\mathcal{G}(\mathcal{L})$ is labeled with $x \leq y$ and $x+y \leq 2$ and has the residue $\langle 1,1,2 \rangle = \langle 1,-1,0 \rangle * \langle 1,1,2 \rangle$ which represents the residue lequality $x+x \leq 2$.
We observe that although $\mathcal{L}$ is unsatisfiable, $\mathcal{G}(\mathcal{L})$ has no infeasible simple loop! This motivates the following notation:

Define a *closure* $\mathcal{G}_C(\mathcal{L})$ of the graph $\mathcal{G}(\mathcal{L})$ of a set $\mathcal{L}$ of linear lequalities in a constructive way: For each admissible simple loop $\mathcal{P}$ (modulo cyclic permutation and reversal) of $\mathcal{G}(\mathcal{L})$ add a new edge labeled with the residue lequality $l(\mathcal{P})$.
The graph $\mathcal{G}_C(\mathcal{L})$ is also called a *closed graph* for $\mathcal{L}$.
If the graph $\mathcal{G}(\mathcal{L})$ of a set $\mathcal{L}$ of linear lequalities is closed, $\mathcal{L}$ is called a *closed set*. If furthermore the graph $\mathcal{G}(\mathcal{L})$ does not contain a simple equality loop, $\mathcal{L}$ is called a *strictly closed set*.

The reverse of an admissible (simple) loop is always admissible, and the cyclic permutations of an admissible (simple) loop are admissible if and only if a_1 and b_n have opposite signs where $\langle a_1,b_1,c_1 \rangle$, $\langle a_2,b_2,c_2 \rangle$, ... $\langle a_n,b_n,c_n \rangle$ is the triple sequence of this path.
Furthermore, all cyclic permutations of an infeasible permutable (simple) loop are infeasible and the reverse of an infeasible (simple) loop is infeasible.[10] This justifies the restriction modulo cyclic permutation and reversal in the previous definition.
Note that the closure is not uniquely defined because the simple admissible loops are chosen modulo cyclic permutation and reversal.
In the example, a graph $\mathcal{G}_C(\{x \geq 2,\ x \leq y,\ x+y \leq 2\})$ is obtained by adding to $\mathcal{G}(\mathcal{L})$ an edge labeled with $0v_0+2x \leq 2$ or, equivalently, $x \leq 1$. This results with lequality $-x+v_0 \leq -2$ (or $x \geq 2$) in an infeasible simple loop labeled with the lequalities $0v_0+2x \leq 2$ and $-x+v_0 \leq -2$ and with residue $\langle 0,0,-2 \rangle$.

Theorem 2.3.
If $\mathcal{G}$ is a closed graph for a set $\mathcal{L}$ of linear lequalities, then the following statements are equivalent:

i) $\mathcal{L}$ is satisfiable
ii) $\mathcal{G}$ has no infeasible simple loop

Proof.
Because of its length, the proof is omitted here. But since this main theorem is fundamental for the new procedure of the extended class, it is given in appendix A. □

10: These statements are proved in appendix A.

2.2.3. The Loop-Residue Procedure

Based on Theorem 2.3, we now present a decision procedure in figure 2.1, the LOOP_RESIDUE procedure, for the satisfiability of a set $\mathcal{L}$ of linear lequalities of the unextended class by computing loop residues. Such a set $\mathcal{L}$ represents a conjunction of atomic formulas.
Since there are finitely many admissible simple loops both in the graph $G(\mathcal{L})$ and in a closed graph $G_C(\mathcal{L})$, the procedure terminates. Its soundness and completeness straightforwardly follow from Theorem 2.3.
Observe that this procedure only decides the satisfiability of a set of linear lequalities. It does not construct a solution for a satisfiable set. This can be done according to the proof of the main theorem in appendix A by constructing a solution for a set $\mathcal{L}$ from a closure $G_C(\mathcal{L})$ without an infeasible simple loop. But for applications such as program verification, one is interested only in satisfiability; knowledge of an explicit solution is not required in this case.

(0) *For a set $\mathcal{L}$ of linear lequalities, construct the graph $G=G(\mathcal{L})$*
(1) *Enumerate the simple admissible loops of G*
modulo cyclic permutation and reversal,
and compute their residues;
IF *any infeasible simple loop is enumerated*
THEN RETURN ($\mathcal{L}$ is unsatisfiable)
ELSE (2)
(2) *Form a closure of G*
by adding a new edge for each residue lequality;
Compute the residues
of all newly formed admissible simple loops[11]
IF *any loop is infeasible*
THEN RETURN ($\mathcal{L}$ is unsatisfiable)
ELSE RETURN ($\mathcal{L}$ is satisfiable)

Figure 2.1: LOOP_RESIDUE procedure deciding the satisfiability of a set $\mathcal{L}$ of linear lequalities

11: Note also that the new admissible loops formed in (2) must have initial vertex v_0.

The implementation of the procedure requires some means of enumerating the simple loops of a graph. There are several algorithms (by Johnson 1975 [12] , Read and Tarjan 1973 [23], and Szwarcfiter and Lauer 1974 [29]) which operate in time order $l*(|V|+|E|)$, and space order $(|V|+|E|)$, where l is the number of simple loops, $|V|$ is the number of vertices, and $|E|$ is the number of edges. A graph may have exponentially many simple loops (in $|E|$) and therefore the decision procedure shows worst-case exponential behaviour.
Though the worst case complexity is exponential, the average case seems to have a pleasant complexity:
"In practise, however, one does not encounter such behavior. The sets of inequalities that arise from verification conditions usually have the form of transitivity chains. The corresponding graphs are treelike, seldom having more that a few loops. Most of the loops that do occur are 2-loops, which are easily tested at the time the graph is constucted." [Shostak, 27]
So the run time behaviour can be compared with algorithms such as the Simplex-Algorithm which also has worst case exponential time complexity, but is widely used in practise.

2.2.4. Generalizations

Recall that the skolemization of the disjunctive normal form of a negated formula of the quantifier-free Presburger Arithmetic consists of conjunctions of atomic formulas which are either lequalities ($\leq$), strict lequalities ($<$), equalities ($=$), or inequalities ($\neq$) each of which can be written in the form $a_1x_1 + a_2x_2 + \ldots + a_nx_n \ \mathit{rel}\ a_0$ where $\mathit{rel} \in \{\leq, <, =, \neq\}$.
While $a_1x_1 + a_2x_2 + \ldots + a_nx_n = a_0$ can be replaced by the conjunction of $a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq a_0$ and $-a_1x_1 - a_2x_2 - \ldots - a_nx_n \leq -a_0$, neither a strict lequality nor an inequality of the form $a_1x_1 + a_2x_2 + \ldots + a_nx_n \ \mathit{rel}\ a_0$ where $\mathit{rel} \in \{<, \neq\}$ can be expressed by lequalities. We therefore require the extension of the LOOP_RESIDUE procedure to strict lequalities. With strict lequalities, $a_1x_1 + a_2x_2 + \ldots + a_nx_n \neq a_0$ can be replaced by the disjunction of $a_1x_1 + a_2x_2 + \ldots + a_nx_n < a_0$ and $-a_1x_1 - a_2x_2 - \ldots - a_nx_n < -a_0$.
So far, we have restricted the LOOP_RESIDUE procedure to decide the satisfiability of a set with at most two variables per lequality. A generalized version of the LOOP_RESIDUE procedure for lequalities and strict lequalities with an arbitrary number of variables is presented in figure 2.3. These two generalizations were proposed by Shostak in [27].

2.2.4.1. Strict lequalities

The version of the LOOP_RESIDUE procedure presented in figure 2.1 is restricted to conjunctions of lequalities, i.e. atomic formulas with the relation $\leq$. Lemma 2.2 indicates how the procedure is generalized to handle strict lequalities of the form $ax + by < c$:
Let us denote by a *strict path* $\mathcal{P}$ such a path with at least one strict lequality and by $\mathcal{L}_\mathcal{P}$ the set of the lequalities and strict lequalities labeling $\mathcal{P}$.
Then, for an admissible simple strict path $\mathcal{P}$, the sets $\mathcal{L}_\mathcal{P}$ and $\mathcal{L}_{\mathcal{P}<}$ are equivalid. This can similiarly be proved as in Lemma 4.2.
By Lemma 2.2 ii), we have that $\mathcal{L}_{\mathcal{P}<}$ and $\mathcal{L}_{\mathcal{P}<} \cup \{l_{\mathcal{P}<}\}$ are equivalid where $l_{\mathcal{P}<}$ is the strict lequality $a(\mathcal{P})x + b(\mathcal{P})y < c(\mathcal{P})$.
Therefore, $\mathcal{L}_\mathcal{P}$ and $\mathcal{L}_\mathcal{P} \cup \{l_{\mathcal{P}<}\}$ are equivalid for a strict path $\mathcal{P}$.

An admissible strict loop $\mathcal{P}$ is *infeasible* if and only if $a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) \leq 0$. The *residue strict lequality* $l_\mathcal{P}$ of an admissible strict path $\mathcal{P}$ is the strict lequality $a(\mathcal{P})x + b(\mathcal{P})y < c(\mathcal{P})$ where x and y are the first and last vertices, respectively, of $\mathcal{P}$.
Including this extension to strict lequalities, Theorem 2.3 still holds for a set $\mathcal{L}$ of linear lequalities and strict lequalities, as noted by Shostak [27].

2.2.4.2. Linear lequalities with an arbitrary number of variables

The LOOP_RESIDUE procedure in figure 2.1 is for convenience designed to decide the satisfiability of a set of linear lequalities with at most two variables per lequality.
A further generalization includes linear (strict) lequalities with an arbitrary number of variables. This can be done by symbolic computation as illustrated in the following example:
Consider the set $\mathcal{L} = \{x \leq y,\ z \leq y - x + 1,\ x \geq 2\}$.

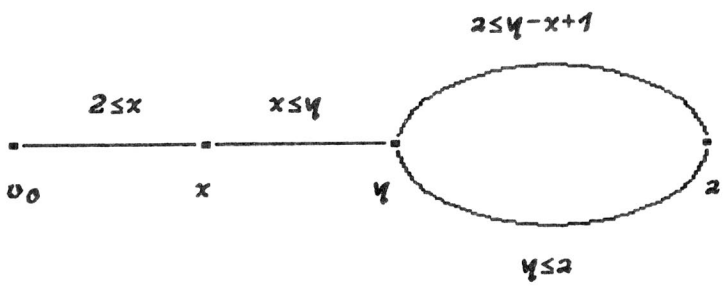


Figure 2.2: Graph $\mathcal{G}$ for $\mathcal{L} = \{x \leq y,\ z \leq y - x + 1,\ x \geq 2\}$

One lequality ($z \leq y - x + 1$) of the set $\mathcal{L}$ has three variables two of which (y, z) are chosen as endpoints of the corresponding edge in the graph $\mathcal{G} = \mathcal{G}(\mathcal{L})$. Symbolic computation in the only admissible simple loop (yzy) yields the residue $\langle -1,1,-x+1 \rangle * \langle -1,1,0 \rangle = \langle -1,1,-x+1 \rangle$ with residue lequality $-y + y \leq -x + 1$ or $x \leq 1$.
Adding this lequality to the graph results with the lequality $x \geq 2$ in an infeasible simple loop ($v_0 x v_0$), thus showing the unsatisfiability of $\mathcal{L}$.
The procedure in figure 2.3 assumes an ordering of the variables other than v_0. The two lowest ranked variables of a (strict) lequality are called *primary variables*. A (strict) lequality with more than two variables labels an edge which is attached to the two nodes corresponding to the primary variables of the (strict) lequality. The other lequalities are treated as usual. The procedure for deciding the satisfiability of a set of (strict) lequalities with an arbitrary number of variables is given in figure 2.3.
Since the number of nonprimary variables decreases in each iteration, the procedure must terminate.
As noted by Shostak [27], the generalized procedure is complete as well as sound which can be proved as an extension of the main theorem.

```
RESET(repeatstop)
REPEAT
  Compute a closure G_C(L) for the set L
  using symbolic evaluation for the residues
  IF G_C(L) has an infeasible loop
    THEN RETURN (L is unsatisfiable)
    ELSIF not all the variables of L are primary
        THEN L := residue (strict) lequalities (G_C(L))
        ELSE SET(repeatstop)
UNTIL repeatstop
RETURN (L is satisfiable)
```

Figure 2.3: LOOP_RESIDUE procedure for a set $\mathcal{L}$ of linear (strict) lequalities with an arbitrary number of of variables per (strict) lequality.

3. Quantifier-free theory of equality

The language of the quantifier-free theory of equality is a subclass of the first-order predicate logic. The only predicate symbol is the equality symbol =. Unlike the quantifier-free Presburger Arithmetic, the quantifier-free theory of equality contains uninterpreted function symbols.
The quantifier-free theory of equality is described in section 3.1. Given a graph and a relation on its vertices, we then define the notion of the congruence closure and give examples in sections 3.2 and 3.3, respectively.
Section 3.4 points out the relation of the congruence closure to the decision problem of the quantifier-free theory of equality and section 3.5 presents a procedure for computing the congruence closure.
This procedure was presented in the paper "Fast Decision Procedures Based on Congruence Closure" of Nelson and Oppen [17]. The basic theorem is given in Shostak's paper "An Algorithm for Reasoning About Equality" [24].

3.1. The theory

We first introduce the notion of a term:

Given a set $\mathcal{U}$ of variables and a family $(\mathcal{F}_i)_{i\geq 0}$ of function symbols where each function symbol $f \in \mathcal{F}_n$ has arity n, we define the set $\mathcal{T}_0$ of *terms* by the smallest set which satisfies

i) $\mathcal{U} \subseteq \mathcal{T}_0$, and

ii) for any number $n \geq 0$, any function symbol $f \in \mathcal{F}_n$, and any terms $t_i \in \mathcal{T}_0$, $1 \leq i \leq n$, $\quad f(t_1, \ldots, t_n) \in \mathcal{T}_0$.

The set $\mathcal{T}$ of *ground terms* is defined by the smallest set such that for any number $n \geq 0$, any function symbol $f \in \mathcal{F}_n$, and any terms $t_i \in \mathcal{T}$, $1 \leq i \leq n$,

$$f(t_1, \ldots, t_n) \in \mathcal{T}.$$

The function symbols with arity 0 are also called constants.

The set of formulas of the quantifier-free theory of equality is the smallest set satisfying the following conditions:

i) any equation $t=t'$ is an (atomic) formula of this set where $t, t' \in \mathcal{T}_0$

ii) for any two formulas $\mathcal{F}$ and $\mathcal{F}'$ of the quantifier-free theory of equality, the formulas $\neg\mathcal{F}$, $\mathcal{F} \vee \mathcal{F}'$, $\mathcal{F} \wedge \mathcal{F}'$, and $\mathcal{F} \supset \mathcal{F}'$ are also formulas of the quantifier-free theory of equality.

A formula of the quantifier-free theory of equality can be transformed into

a formula of the quantifier-free theory of equality without function symbols by a procedure which is given in figure 4.1. The resulting formula is also contained in the quantifier-free Presburger Arithmetic. The quantifier-free theory of equality is thus decidable.

3.2. Congruence closure

Let $G=(V,E)$ be a directed graph with labeling function l which assigns a function symbol to each vertex. For a directed edge from u to v, u is called predecessor of v, and v is called successor of u. Let the edges leaving a vertex v be ordered and denote by $v[i]$, $1 \le i \le n(v)$, the i-th successor of v wrt the i-th edge leaving v, where $n(v)$ denotes the outdegree of vertex v, i.e., the number of edges leaving v. Multiple edges are allowed, i.e., $v[i]=v[j]$ is possible for $i \neq j$.

Let $R \subseteq V \times V$ be a binary relation on V. Define the relation $C_R \subseteq V \times V$ by $(u,v) \in C_R$ if and only if $l(u)=l(v)$, $n(u)=n(v)$, and $(u[i], v[i]) \in R$ for all $1 \le i \le n(u)$.

The pairs of vertices in C_R are said to be *congruent* under R. R is *closed under congruences* if and only if $C_R \subseteq R$.

Define the *congruence closure* R_C of R as the minimal extension of R such that R_C is an equivalence relation and closed under congruences.

One can prove that the congruence closure of a relation is uniquely defined.

3.3. Examples

Before the connection of the decision problem of the quantifier-free theory of equality with the notion of the congruence closure of a relation on a graph is pointed out in section 3.4 and an algorithm for computing the congruence closure is given in section 3.5, let us first consider the following two examples:

Let G_1 be the graph shown in figure 3.1, and let $R = \{(v2, v3)\}$.

Since the congruence closure R_C of a relation R is an equivalence relation, we can represent R_C by its corresponding partion Π.

The partition $\Pi_0 = \{\{v_1\}, \{v_2, v_3\}, \{v_4\}\}$ is too fine, since we have $(v_1, v_2) \in C_R$. So we must merge the equivalence classes of the vertices v_1 and v_2. We obtain the partition $\Pi_1 = \{\{v_1, v_2, v_3\}, \{v_4\}\}$. This partition represents an equivalence relation which is closed under congruences. It thus represents the congruence closure of the relation R.

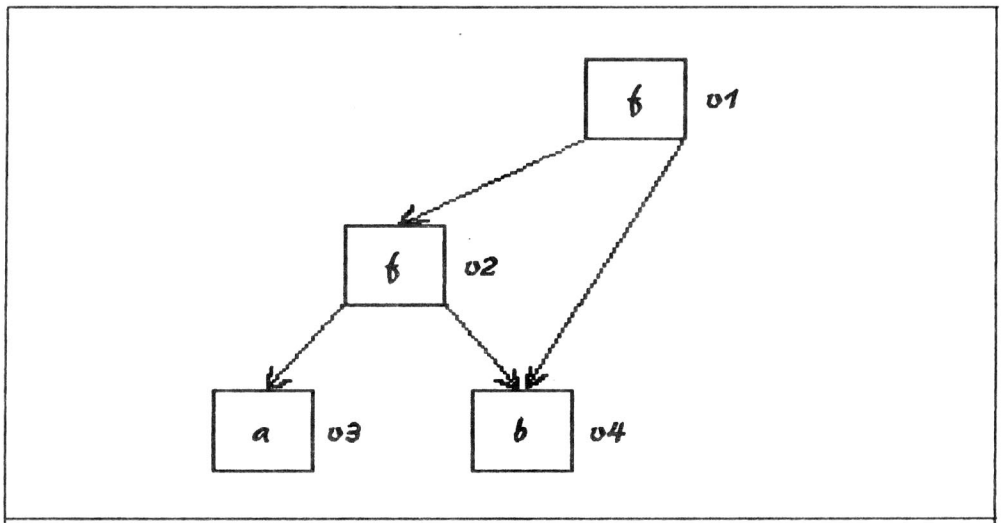


Figure 3.1: Graph G_1

As another example, consider the graph G_2 in figure 3.2 and the relation $\mathcal{R} = \{(v1, v6), (v3, v6)\}$.
Starting with the partition $\Pi_0 = \{\{v_1, v_3, v_6\}, \{v_2\}, \{v_4\}, \{v_5\}\}$ which corresponds to the smallest equivalence relation containing $\mathcal{R}$, we construct a sequence of partitions by merging the equivalence classes of congruent[12] vertices in order to obtain the congruence closure of $\mathcal{R}$.
Since $(v_3, v_6) \in \mathcal{R}(\Pi_0)$, the vertices v_2 and v_5 are congruent under $\mathcal{R}(\Pi_0)$. So we merge the equivalence classes of v_2 and v_5 and obtain $\Pi_1 = \{\{v_1, v_3, v_6\}, \{v_2, v_5\}, \{v_4\}\}$. Now, since $(v_2, v_5) \in \mathcal{R}(\Pi_1)$, the vertices v_1 and v_4 are congruent under $\mathcal{R}(\Pi_1)$. And, finally, for $\Pi_2 = \{\{v_1, v_3, v_4, v_6\}, \{v_2, v_5\}\}$, the vertices v_3 and v_5 are congruent under $\mathcal{R}(\Pi_2)$, since $(v_4, v_6) \in \mathcal{R}(\Pi_2)$. So Π_3 consists of one equivalence class. The congruence closure of $\mathcal{R}$ is thus $\mathcal{R}_C = \{v_1, v_2, v_3, v_4, v_5, v_6\}$[2].

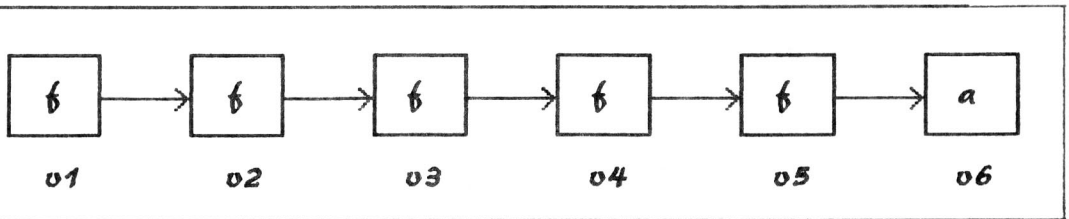


Figure 3.2: Graph G_2

12: Congruent wrt the (equivalence) relation $\mathcal{R}(\Pi)$ which is associated with the partition Π.

3.4. Congruence closure and the quantifier-free theory of equality

In order to point out the connection between the notion of the congruence closure of a binary relation represented by a graph and the decision problem of the quantifier-free theory of equality, we assign a term $t(v)$ to each vertex v recursively in the following way:
For a vertex v without successor let $t(v)=l(v)$ and for a vertex v with outdegree $n=n(v)$ let $t(v) = l(v)\ (t(v[1]), \ldots, t(v[n]))$.
Denote in addition for a relation $\mathcal{R}$ by $\mathcal{T}_{\mathcal{R}}$ the set $\mathcal{T}_{\mathcal{R}} = \{\ t(v)=t(v')\ |\ (v,v') \in \mathcal{R}\ \}$. It now can be proved that for a relation $\mathcal{R}$ and its closure $\mathcal{R}_C$, $\mathcal{T}_{\mathcal{R}}$ and $\mathcal{T}_{\mathcal{R}_C}$ are equivalid.

For the first example, $\mathcal{T}_{\mathcal{R}} = \{\ f(a,b)=a\ \}$ and $f(f(a,b),b) = a \in \mathcal{T}_{\mathcal{R}_C}$.
One could prove, for example, the validity of the formula $\mathcal{F} \equiv f(a,b)=a \supset f(f(a,b),b) = a$ in the following way:
Negate the formula and place it into disjunctive normal form. We obtain for the negation of $\mathcal{F}$ the two disjunctions $f(a,b)=a$ and $f(f(a,b),b) \neq a$. The congruence closure is now computed for the equalities and finally for each inequality $t \neq t'$, it is tested if the two terms t and t' belong to the same equivalence class. If one such inequality exists, then the negation of the formula $\mathcal{F}$ is unsatisfiable and thus $\mathcal{F}$ is valid. Otherwise, $\mathcal{F}$ is not valid.
Since in our example the terms $f(f(a,b),b)$ and a are contained in the same equivalence class of the congruence closure, $\mathcal{F} \equiv f(a,b)=a \supset f(f(a,b),b) = a$ is a valid formula.

And for the second example, $\mathcal{T}_{\mathcal{R}} = \{\ fffffa = a,\ fffa=a\ \}$ and $fa=a \in \mathcal{T}_{\mathcal{R}_C}$.
Similiarly, the formula $fffffa = a \wedge fffa=a \supset fa=a$ can be proved as a valid formula.

These examples indicate that the concepts of rewriting might provide useful support for such decision procedures. The concepts are introduced in section 4.4 and it indeed turns out that they are very useful.

3.5. An algorithm for the congruence closure

We give an algorithm for computing the congruence closure according to the paper of Nelson and Oppen [17] which is shown in figure 3.3.
As in the examples, an equivalence relation is represented by its corresponding partition.

We now argue that the procedure MERGE is correct in the following sense:

```
CONGRUENCE_CLOSURE ( R ):
        E := { E_v | E_v = {v} for all v ∈ V }
        FOR ALL (u,v) ∈ R DO MERGE (u,v)

MERGE(u,v):
        IF E_u ≠ E_v
          THEN
            BEGIN
              P_u := {u_p | ∃ u_e. u_p is predecessor of u_e
                        and u_e is equivalent to u (i.e., u_e ∈ E_u) }
              P_v := {v_p | ∃ v_e. v_p is predecessor of v_e
                        and v_e is equivalent to v (i.e., v_e ∈ E_v) }
              E := (E - {E_u, E_v}) ∪ {E_u ∪ E_v}
              FOR ALL (x,y) ∈ P_u × P_v DO
                  IF E_x ≠ E_y AND CONGRUENT(x,y)
                    THEN MERGE(x,y)
            END

CONGRUENT(x,y):
        IF l(x)=l(y) AND n(x)=n(y)
          THEN BEGIN
                  FOR i:=1 TO n(x) DO
                      IF E_x[i] ≠ E_y[i]
                        THEN RETURN FALSE
                  RETURN TRUE
               END
          ELSE RETURN FALSE
```

Figure 3.3: Procedure CONGRUENCE_CLOSURE with subprocedures MERGE and CONGRUENT

If a partition $\mathcal{E}$ represents the congruence closure of a relation $\mathcal{R}$, then MERGE(u,v) constructs the congruence closure of the relation $\mathcal{R} \cup \{(u,v)\}$: Since the equivalence classes of vertices u and v are only merged for pairs (u,v) of the relation $\mathcal{R}$ by external calls of the procedure MERGE or for

congruent pairs by internal calls, the resulting equivalence relation is not too coarse.
Suppose it is too fine. Then there are vertices x and y, such that they are congruent but not equivalent. Since $\mathcal{R}$ was closed under congruences, x and y were not congruent initially. Then there must have been some call of the procedure MERGE for some vertices a and b such that either (x,y) or (y,x) is in $\mathcal{P}_a \times \mathcal{P}_b$. In accordance to the algorithm, either MERGE(x,y) or MERGE(y,x) is called which merges the equivalence classes of x and y and thus makes x and y equivalent contrary to the assumption that the equivalence relation is too fine.

Therefore, if the relation represented by $\mathcal{E}$ is closed under congruences before calling the procedure MERGE, then it is also closed under congruences afterwards. The procedure CONGRUENCE_CLOSURE thus computes the congruence closure of a relation $\mathcal{R}$.
In their paper, Nelson and Oppen show furthermore that the algorithm for computing the congruence closure can be implemented in $O(m^2)$ worst case time behaviour and a sophisticated version in $O(m*\log(m))$ where m is the number of edges in the graph. It is assumed that there are no isolated vertices, i.e., $n=O(m)$ where n is the number of vertices.

4. Quantifier-free Presburger Arithmetic extended by predicate and function symbols

The quantifier-free Presburger Arithmetic extended by predicate and function symbols includes, in addition to the quantifier-free Presburger Arithmetic, formulas with predicate and function symbols.
The atomic formulas of the unextended class $\mathcal{PA}$ are linear in variables. But the atomic formulas of the extended class $\mathcal{PA}_f$ are linear in terms that have an outermost uninterpreted function symbol. The atomic formulas are furthermore augmented by such formulas as $\mathcal{P}(t_1, t_2, \ldots, t_n)$ where $\mathcal{P}$ is an uninterpreted predicate symbol and $t_1, t_2, \ldots, t_n$ are terms.
The decidability of $\mathcal{PA}_f$ is established in section 4.1 and a decision procedure for the domain of integer numbers, designed by Shostak [25], is presented in section 4.2.
Based on the given procedure, we look out for new concepts in order to improve the decision process proposed by Shostak in section 4.3. For example, separation of atomic formulas of the form $at + bt' \; rel \; c$ where $rel \in \{\leq, =, \neq\}$ subject to the predicate rel serves for compactness in the representation of the information.
Extracting information, for example by inferring equalities, is an often stated goal. Very helpful is the notion of an equality loop. Equality loops are used to generate equalities from a set of lequalities. This and some more very interesting results are proved in section 4.5.
Considering the connection to rewriting on ground terms, we are able to incorporate recent (1988) results on ground rewriting, presented in section 4.4, in the newly developed decision procedure of this diploma thesis, the EQUALITY_LOOP_RESIDUE procedure, given in section 4.6.
Examples indicate how this procedure improves the decision process and in another section, further suggestions are made how the presented method can serve for even more advantageous computation:
The LOOP_RESIDUE procedure in section 2.2 for the unextended class and the EQUALITY_LOOP_RESIDUE procedure in section 4.6 for the extended class were originally designed to be applied to the conjunctions in the disjunctive normal form of the negation of a formula as last step in the decision process of the validity of a formula. But furthermore, they can advantageously be applied to formulas before they are negated and expanded into disjunctive normal form. This holds particularly for the EQUALITY_LOOP_RESIDUE procedure.

An example demonstrates that the application of this procedure to subconjunctions of a given formula $\mathcal{F}$ can substantially reduce the number of conjunctions in the disjunctive normal form of the negation by constructing a formula $\mathcal{F}'$ which is equivalent to the formula $\mathcal{F}$ and has much less conjunctions in the disjunctive normal form of its negation than $\mathcal{F}$. Again, an important aspect is the compactness of the formula $\mathcal{F}'$. Especially this is supported by the procedure.

4.1. Decidability

We now prove the decidability of the extended class $\mathcal{PA}_f$ by reducing the problem to the decidability of the unextended class $\mathcal{PA}$:
The procedure in figure 4.1 reduces a formula $\mathcal{F}_{fP}$ in $\mathcal{PA}_f$ to a formula $\mathcal{F}$ in $\mathcal{PA}$. The reduction is carried out in two steps:
First, the predicate symbols are eliminated by introducing new function

(1) *For each n-ary predicate symbol* P [13] *occurring in* $\mathcal{F}_{fP}$ *let* f_P *be a new n-ary function symbol.*
Obtain $\mathcal{F}_f$ *from* $\mathcal{F}_{fP}$ *by replacing each atomic formula* $P(t_1,\ldots,t_n)$ *by the formula* $f_P(t_1,\ldots,t_n)=0$.

(2) *For each pair* $f(t_1,\ldots,t_n)$, $f(u_1,\ldots,u_n)$ *of distinct terms or subterms of terms in* $\mathcal{F}_f$ *with the same outermost function-symbol* f [13], *construct the following axiom:*

$$t_1=u_1 \wedge t_2=u_2 \wedge \ldots \wedge t_n=u_n \supset f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n).$$

Let $\mathcal{F}_{Af}$ *be the formula*

$$\mathcal{F}_{Af} \equiv (A_1 \wedge A_2 \wedge \ldots \wedge A_t) \supset \mathcal{F}_f,$$

where the A_i *'s are the axioms so constructed.*

(3) *For each term t occurring in* $\mathcal{F}_{Af}$ *that has an outermost function symbol, let* x_t *be a new variable.*
Obtain $\mathcal{F}$ *from* $\mathcal{F}_{Af}$ *by replacing each such term t by* x_t
(In case where one such term is nested within another, the larger term is replaced).

Figure 4.1: Reduction of a formula $\mathcal{F}_{fP}$ of the extended class

13: Uninterpreted predicate and function symbols of the Presburger Arithmetic; i.e., the predicate symbol < and the function symbols 0, 1, and + are excluded.

symbols and second, the function symbols are eliminated by adding axioms to the formula and finally replacing the terms by new variables.

Consider, for example, the valid formula

$$\mathcal{F}_{fp} \equiv \quad [\, p(x) \supset x=2 \wedge f(2*z)=f(z) \wedge g(y)=x+7 \,] \supset [\, f(g(y))=f(5+2*x) \vee \neg p(x) \,]$$

(either holds $\neg p(x)$ or $p(x)$ implying $x=2$ and $g(y)=x+7=5+2*x$). Substituting the predicate symbol p by f_p, we obtain the formula

$$\mathcal{F}_f \equiv \quad [\, f_p(x)=0 \supset x=2 \wedge f(2*z)=f(z) \wedge g(y)=x+7 \,] \supset [\, f(g(y))=f(5+2*x) \vee \neg(f_p(x)=0) \,].$$

The formula $\mathcal{F}_f$ contains six pairs of distinct terms with the same outermost function symbol - $\binom{4}{2}$ pairs among the terms $f(2*z)$, $f(z)$, $f(g(y))$, and $f(5+2*x)$. Therefore, the formula $\mathcal{F}_{Af}$ is given by

$$\begin{aligned}\mathcal{F}_{Af} \equiv \quad \{ \quad & 2*z=z \supset f(2*z)=f(z) \\ \wedge \quad & 2*z=g(y) \supset f(2*z)=f(g(y)) \\ \wedge \quad & 2*z=5+2*x \supset f(2*z)=f(5+2*x) \\ \wedge \quad & z=g(y) \supset f(z)=f(g(y)) \\ \wedge \quad & z=5+2*x \supset f(z)=f(5+2*x) \\ \wedge \quad & g(y)=5+2*x \supset f(g(y))=f(5+2*x) \quad \} \\ \supset \quad & \{\, [\, f_p(x)=0 \supset x=2 \wedge f(2*z)=f(z) \wedge g(y)=x+7 \,] \\ \supset \quad & [\, f(g(y))=f(5+2*x) \vee \neg(f_p(x)=0) \,] \,\}.\end{aligned}$$

We obtain $\mathcal{F}$ by replacing the terms $f_p(x)$, $f(2*z)$, $f(z)$, $g(y)$, $f(g(y))$ and $f(5+2*x)$ by the variables v_1, v_2, v_3, v_4, v_5, and v_6:

$$\begin{aligned}\mathcal{F} \equiv \quad \{ \quad & 2z=z \supset v_2=v_3 \\ \wedge \quad & 2z=v_4 \supset v_2=v_5 \\ \wedge \quad & 2z=5+2*x \supset v_2=v_6 \\ \wedge \quad & z=v_4 \supset v_3=v_5 \\ \wedge \quad & z=5+2*x \supset v_3=v_6 \\ \wedge \quad & v_4=5+2*x \supset v_5=v_6 \quad \} \\ \supset \quad & \{\, [\, v_1=0 \supset x=2 \wedge v_2=v_3 \wedge v_4=x+7 \,] \\ \supset \quad & [\, v_5=v_6 \vee \neg(v_1=0) \,] \,\}.\end{aligned}$$

The formula $\mathcal{F}$ is contained in the unextended class, and therefore its validity can be decided.
"The reduction just described is quite similiar to Ackermann's [1] method for eliminating function symbols from universally quantified equality formulas in predicate calculus with function symbols and identity. The correctness of the reduction can be proved straightforwardly; given a model for $\neg\mathcal{F}_{fp}$, one can construct a model for $\neg\mathcal{F}$, and conversely. The details are easily gleaned from Ackermann's proof, and so are omitted here." [25]
Now since $\neg\mathcal{F}_{fp}$ is unsatisfiable if and only if $\neg\mathcal{F}$ is unsatisfiable, the formulas $\mathcal{F}_{fp}$ and $\mathcal{F}$ are equivalid proving the decidability of the extended class.
Observe that not only the two formulas $\mathcal{F}_{fp}$ and $\mathcal{F}$ are equivalid, but also $\mathcal{F}_{fp}$, $\mathcal{F}_{f}$, $\mathcal{F}_{Af}$, and $\mathcal{F}$ are all equivalid formulas.

4.2. A practical decision procedure

This section presents a decision procedure for the extended class with integer domain which has been designed by Shostak [25]. It introduces to the main problem of the extension by function symbols and shows by an example that the difference between the number of conjunctions generated in the reduction process and the number of conjunctions necessary to decide the validity of a formula tends to grow enormously.

4.2.1. The domain

Shostak has chosen as domain the set of integer numbers.
The formula $\mathcal{F}$ to be decided is reduced to a set of integer linear programming problems (ILP's) C_i such that $\neg\mathcal{F}$ and $C_1 \vee C_2 \vee \ldots \vee C_p$ are equivalid for the integer domain. This is done by negating $\mathcal{F}$ and expanding the negation into disjunctive normal form such that each C_i is a conjunction of linear lequalities of the form $A \leq B$. This is achieved by replacing formulas of the form $A=B$, $A \geq B$, $A<B$, and $\neg(A \leq B)$ by the formulas $(A \leq B \wedge B \leq A)$, $B \leq A$, $A+1 \leq B$, and $B+1 \leq A$, respectively, during the reduction process. Note that this replacement is correct since the domain is the set of integer numbers.

4.2.2. Semantic notations

We now need some definitions that refer to the semantic:
Given a set $\mathcal{V}$ of variables, a family $(\mathcal{F}_i)_{i \geq 0}$ of function symbols, and a

family $(\mathcal{P}_i)_{i\geq 0}$ of predicate symbols, an *interpretation* φ wrt a domain $\mathcal{D}$ is an assignment such that $\varphi(v)$ is an element of $\mathcal{D}$ for a variable v, $\varphi(f)$ is a function $\mathcal{D}^n \rightarrow \mathcal{D}$ for a function symbol f in accordance to its arity, and $\varphi(\mathcal{P})$ is a subset of $\mathcal{D}^n$ for an n-ary predicate symbol $\mathcal{P}$.
An interpretation is extended to terms in a natural way as a homomorphism: to each term $t = f(t_1, \ldots, t_n)$, it assigns $\varphi(f)(\varphi(t_1), \ldots, \varphi(t_n))$.
Analogously, the atomic formulas $\mathcal{P}(t_1, t_2, \ldots, t_n)$ are interpreted in such a way that $\varphi(\mathcal{P}(t_1, t_2, \ldots, t_n))$ if and only if $(\varphi(t_1), \varphi(t_2), \ldots, \varphi(t_n)) \in \varphi(\mathcal{P})$.
The extension to all quantifier-free formulas is done by interpreting the boolean connectives $\neg$, $\vee$, $\wedge$, and $\supset$ as usual.

An interpretation φ of a formula of the Presburger Arithmetic additionally satisfies $\varphi(<) = <$ for the interpreted predicate symbol $<$, $\varphi(0) = 0$, $\varphi(1) = 1$, and $\varphi(+) = +$ for the interpreted function symbols, and the axioms of the Presburger theory.

We now introduce a notation for modifying an interpretation φ:
Given a variable v and an element ∂ of the domain $\mathcal{D}$, we denote by $\varphi[v \rightarrow \partial]$ that interpretation obtained from φ by assigning the element ∂ to the variable v.

The notion of well definedness which normally refers to a function, is now for convenience extended to function symbols, formulas, and sets:
A function symbol f is said to be *well defined* wrt an interpretation φ, if $\varphi(f)$ is a well defined function.
A *formula* $\mathcal{F}$ is said to be *well defined* wrt an interpretation φ, if $\varphi(f)$ is a well defined function wrt φ for any function symbol f occurring in $\mathcal{F}$.
A *set* $\mathcal{L}$ is said to be *well defined* wrt an interpretation φ, if $\varphi(f)$ is a well defined function wrt φ for any function symbol f occurring in $\mathcal{L}$.

A *solution* φ for a set $\mathcal{L}$ of formulas wrt the domain $\mathcal{D}$ is an interpretation which satisfies all formulas of $\mathcal{L}$.
A set $\mathcal{L}$ of formulas is *satisfiable* wrt a domain $\mathcal{D}$ if and only if there exists a solution φ for the system $\mathcal{S}$ wrt the domain $\mathcal{D}$.
Two sets $\mathcal{L}_1$ and $\mathcal{L}_2$ are said to be *equisatisfiable* if and only the following condition holds:
The set $\mathcal{L}_1$ is satisfiable if and only if the set $\mathcal{L}_2$ is satisfiable.

4.2.3. The basic problem

The reduction described in figure 4.1 does not of itself provide an efficient decision procedure. The example in section 4.1 gives us the intuition that

the reduction is prone to combinatorial explosion. Let us now investigate the reason. Since a predicate symbol may be considered as a special function symbol, it suffices to observe the extension of the language class by introduction of function symbols.

Note that an axiom is constructed for each pair of terms with the same outermost function symbol. The axioms state that the function symbols have to be interpreted by *well defined* functions: A function assigns a uniquely defined value to every argument of its domain. The axioms, which are added as hypotheses to the formula, justify the substitution of terms with an outermost (uninterpreted) function symbol by new variables in the last step of the reduction.

For a function symbol that occurs m times, there are $\binom{m}{2}$ axioms, so the number of axioms is in the worst case proportional to the square of the length of the formula. And in the expansion to disjunctive normal form, an axiom for a n-ary function symbol contributes at least the factor (n+1) to the number of conjunctions, since $t_1=u_1 \wedge t_2=u_2 \wedge \ldots \wedge t_n=u_n \supset f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n)$ is equivalent to the formula $t_1 \neq u_1 \vee t_2 \neq u_2 \vee \ldots \vee t_n \neq u_n \vee f(t_1,\ldots,t_n) = f(u_1,\ldots,u_n)$ which is the disjunction of (n+1) formulas.

To illustrate the combinatorial explosion, consider the formula $\mathcal{F} \equiv (x \leq gx \wedge gx \leq x \supset x=ggggggx)$[14]. There are ten (= $\binom{5}{2}$) pairs of terms among the five subterms gx , ggx , $gggx$, $ggggx$, and $gggggx$. An axiom is constructed for each pair and thus 2^{10}=1024 conjunctions[15] are in the disjunctive normal form of the reduced formula. It is even worse (3^{10}=59049) if $(a<b \vee a>b)$ is used instead of $a \neq b$.

So, obviously, the construction of axioms is responsible for the combinatorial explosion. At this point the question arises how this can be obviated.

Let us look at the formula $\mathcal{F} \equiv (x \leq gx \wedge gx \leq x \supset x=ggggggx)$. Out of the ten axioms, only the four axioms for the pairs $\{gx, ggx\}$, $\{ggx, gggx\}$, $\{gggx, ggggx\}$, and $\{ggggx, gggggx\}$ are relevant. More important, these axioms can be replaced by the single formula $x=gx \supset x=gggggx$.

So one aspect of improvement accounts for distinguishing relevant from irrelevant information and, indeed, in most cases only a few axioms are of relevance. Another aspect is that the relevant information is frequently determinable in advance of its application as shown by the procedure EQPAIRS in the next section.

14: For convenient reading, the parantheses are omitted in this example.

15: The base is 2, because the implications are assumed to be written in the following disjunctive form: $(arg_1 \neq arg_2 \vee g(arg_1)=g(arg_2))$ which uses the relation ≠ to abbreviate $(arg_1<arg_2 \vee arg_1>arg_2)$.

4.2.4. The Procedure

The following approach greatly reduces the number of constructed axioms for the decision of a formula compared with the number of axioms generated in the reduction process.
For a set $\mathcal{G}$ of linear lequalities associate an ILP obtained by replacing each term t with an uninterpreted outermost function symbol by a new variable x_t. The associated ILP thus simply omits the axioms for the function symbols occurring in $\mathcal{G}$ which are constructed in the reduction process given in figure 4.1.
The steps (4.1) and (4.2) of the procedure in figure 4.2 refer to the easier of the cases: no additional formula needs to be constructed and the result for the associated ILP can be transferred to the set $\mathcal{G}_i$ itself.
Now, if a function symbol f in $\mathcal{G}_i$ is not well defined wrt the solution φ for the associated ILP, a formula $\mathcal{H}$ is constructed which summarizes the violation such that $\mathcal{G}$ and $\mathcal{H} \wedge \mathcal{G}$ are equisatisfiable. Step (4) of the procedure is now applied to each conjunction in the disjunctive expansion of $\mathcal{H} \wedge \mathcal{G}$.
As an illustration, let us consider, for example, the formula $\mathcal{F} \equiv (x \leq gx \wedge gx \leq x \supset x = ggggggx)$. The negation expanded into the disjunctive normal form $(\neg\mathcal{F} \equiv \mathcal{G}_1 \vee \mathcal{G}_2)$ is associated with the following two ILP's:

$\mathcal{G}_1 \equiv \{x \leq gx,\ gx \leq x,\ x+1 \leq ggggggx\}$ and $\mathcal{G}_2 \equiv \{x \leq gx,\ gx \leq x,\ ggggggx+1 \leq x\}$

Step (4.1) obtains a solution for the ILP associated with $\mathcal{G}_1$. For example, $x=0$, $gx=0$, and $ggggggx=1$. In this solution, the function symbol g is not interpreted by a well defined function. The violation is expressed by the formula $\mathcal{H} \equiv x=gx \supset gx=ggggggx \equiv \mathcal{H}_1 \vee \mathcal{H}_2 \vee \mathcal{H}_3$ where $\mathcal{H}_1 \equiv x+1 \leq gx$, $\mathcal{H}_2 \equiv gx+1 \leq x$, and $\mathcal{H}_3 \equiv gx=ggggggx$. This formula is now added as an axiom to $\mathcal{F}$ to obtain the formula $\mathcal{H} \supset \mathcal{F}$. If this formula is negated $\neg(\mathcal{H} \supset \mathcal{F}) \equiv (\mathcal{H} \wedge \neg\mathcal{F}) \equiv (\mathcal{H} \wedge \mathcal{G}_1) \vee (\mathcal{H} \wedge \mathcal{G}_2)$ and expanded into disjunctive normal form $(\mathcal{H}_1 \wedge \mathcal{G}_1) \vee (\mathcal{H}_2 \wedge \mathcal{G}_1) \vee (\mathcal{H}_3 \wedge \mathcal{G}_1) \vee (\mathcal{H}_1 \wedge \mathcal{G}_2) \vee (\mathcal{H}_2 \wedge \mathcal{G}_2) \vee (\mathcal{H}_3 \wedge \mathcal{G}_2)$, the resulting six ILP's are found not to have any solutions. Therefore, the original formula $\mathcal{F}$ must be valid.
Note that in this case only seven ILP's (the one associated with $\mathcal{G}_1$ and six to decide the augmented formula $\mathcal{H} \supset \mathcal{F}$) are required to be solved, where in the reduced formula at least 1024 had to be solved. Analyzing this enormous gap, we observe that the reduction process constructs ten axioms six of which are irrelevant and that the four relevant axioms $x=gx \supset gx = ggx$, $gx = ggx \supset ggx = gggx$, $ggx = gggx \supset gggx = ggggx$, and $gggx = ggggx \supset$

$gggg x = ggggg x$ can be replaced by the single formula $x = gx \supset x = ggggg x$ which is done by the procedure EQPAIRS presented in figure 4.3.

(1) *Elimination of predicate symbols:*
For each n-ary predicate symbol $\mathcal{P}$ *occurring in* $\mathcal{F}$, *let* $f_{\mathcal{P}}$ *be a new n-ary function symbol; replace each atomic formula* $\mathcal{P}(t_1,\ldots,t_n)$ *in* $\mathcal{F}$ *by the formula* $f_{\mathcal{P}}(t_1,\ldots,t_n)=0$.

(2) *Elimination of addition in arguments:*
Eliminate addition and multiplication by constants[16] *in arguments by introducing new variables (i.e., replace* $x \leq f(2y+3)$ *by* $y'=2y+3 \supset x \leq f(y')$).

(3) *Negation and construction of a disjunctive normal form:*
$G_1 \vee G_2 \vee \ldots \vee G_p$ *is obtained, where each* G_i *is a conjunction of linear lequalities of the form* $A \leq B$

(4) *All* G_i's *are unsatisfiable if and only if* $\mathcal{F}$ *is valid. Each such* G_i *is now tested for satisfiability in the following way:*

(4.1.) *Solve the ILP associated with* G_i
IF *there is no solution for the associated ILP*
THEN G_i is unsatisfiable
ELSE (4.2.)

(4.2.) IF G_i *is well defined wrt the found solution*[17]
THEN G_i is satisfiable
ELSE (4.3.)

(4.3.) *Summarize the violation in a formula* H, *and reapply step (4) to each conjunction in the disjunctive expansion of* $H \wedge G_i$

Figure 4.2: Decision procedure for the extended theory

The completeness of the procedure for the extended class wrt the integer domain depends on the completeness of the ILP solver in step (4.1).
As noted in the second chapter, the LOOP_RESIDUE procedure for deciding the satisfiability of a set of linear lequalities in variables is not of itself integer-complete[18].

16 : Recall that multiplication by constants is an abbreviation for addition.

17: The solution for the ILP is then also a solution for G_i.

18 : For example, a closed graph for $\{ 1 \leq 2x, 2x \leq 1 \}$ has neither an infeasible simple loop nor an integer solution.

On the other hand, incorporating integer concepts to obtain an integer-complete version of the LOOP_RESIDUE method may result in too inefficient procedures.
We have not yet shown how to examine the well definedness of a set of formulas wrt an interpretation and how to construct a formula that summarizes the violation. For these tasks, Shostak [25] has presented an appropriate recursive procedure, the procedure EQPAIRS given in figure 4.3.

EQPAIRS (t_1, t_2, *alreadytried*) =

IF $\langle t_1, t_2 \rangle \in$ *alreadytried*
THEN RETURN ϕ
ELSIF $t_1 \in E_{t_2}$
THEN RETURN $\{ \langle t_1, t_2 \rangle \}$
ELSIF *for some function symbol* f *and terms* u_1, u_2:
(i) $f(u_1) \in E_{t_1}$, *and*
(ii) $f(u_2) \in E_{t_2}$, *and*
(iii) EQPAIRS $(u_1, u_2,$ *alreadytried* $\cup \{\langle t_1, t_2 \rangle\}) \neq \phi$
THEN RETURN $P_1 \cup P_2 \cup P_3$, *where*
P_1 = IF $t_1 = f(u_1)$ THEN ϕ ELSE $\{\langle t_1, f(u_1)\rangle\}$
P_2 = IF $t_2 = f(u_2)$ THEN ϕ ELSE $\{\langle t_2, f(u_2)\rangle\}$
P_3 = EQPAIRS $(u_1, u_2,$ *alreadytried* $\cup \{\langle t_1, t_2 \rangle\}$
ELSE RETURN ϕ.

Figure 4.3: Procedure EQPAIRS

Let φ be the discovered integer solution for the G_i whose satisfiability is to be determined, and T the set of terms to which φ assigns values. Denote by U the set of all subterms of terms in T. Finally, for each term $t \in U$, define the set E_t by $E_t = \{ t' \in T \mid \varphi(t) = \varphi(t') \}$ for $t \in T$ and $E_t = \{t\}$ otherwise.
The third argument is empty on external calls and is used in internal calls to prevent infinite recursion. EQPAIRS returns a set of pairs in $T \times T$.

Shostak states that φ has no violations of substitutivity if and only if for all $t, t' \in T$, either $\varphi(t)=\varphi(t')$ or EQPAIRS$(t,t',\emptyset)=\emptyset$.
Furthermore, if for some terms $t, t' \in T$, $\varphi(t) \neq \varphi(t')$ and EQPAIRS$(t,t',\emptyset)$ $= \{\langle t_1,s_1\rangle, \ldots, \langle t_n,s_n\rangle\}$, $n \geq 1$, then the formula

$$\mathcal{H} \equiv [t_1=s_1 \wedge \ldots \wedge t_n=s_n \supset t=t']$$

follows from substitutivity but is not satisfied by φ.
To check φ for violations of substitutivity, it thus suffices to compute EQPAIRS$(t,t',\emptyset)$ for such pairs t, t' of terms in T to which different values are assigned by φ. A violation exists if and only if EQPAIRS$(t,t',\emptyset)$ $\neq \emptyset$ for some such pair. In such a case, the formula $\mathcal{H}$ summarizes the violation [25].
Note that the procedure is called for terms t and t' with *same outermost uninterpreted function symbol*, but different assigned values. It tests whether terms are congruent using the notation of the previous chapter.
The congruence is based on a relation which is induced by the solution φ of the associated ILP and on the well-definedness of the functions symbols wrt to the interpretation φ.
This shows the connection between the quantifier-free theory of equality and the well-definedness. For further information and for the termination of the procedure in figure 4.2, the interested reader is referred to the paper of Shostak [25]. Though there can also be found some suggestions for improvements, they do not change the structure of the procedure.
Instead of examining these here again, let us search for new concepts in the next section which may lead to an essential improvement.

4.3. New concepts

Recall that the LOOP_RESIDUE procedure in section 4.2 constructs linear lequalities from the atomic formulas in the conjunctions to be decided for satisfiability by replacing formulas of the form $\mathcal{A}=\mathcal{B}$, $\mathcal{A} \geq \mathcal{B}$, $\mathcal{A}<\mathcal{B}$, and $\neg(\mathcal{A} \leq \mathcal{B})$ by the formulas $(\mathcal{A} \leq \mathcal{B} \wedge \mathcal{B} \leq \mathcal{A})$, $\mathcal{B} \leq \mathcal{A}$, $\mathcal{A}+1 \leq \mathcal{B}$, and $\mathcal{B}+1 \leq \mathcal{A}$, respectively, during the reduction process.
This simplifies the decision procedure because there is only one sort of atomic formulas to be treated.
On the other hand, compact information given by the relations = and $\neq$ is splitted up into a conjunction or disjunction of two atomic formulas.

Let us now examine an instructive example.
Consider the formula

$$\mathcal{F} \equiv \{ (f(a,c) \le a \wedge a \le f(a,c)) \vee (x \le g(x) \wedge g(x) \le x) \vee (b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b) \} \supset \{ (f(f(a,c),c) \le a \wedge a \le f(f(a,c),c)) \vee (g^5(x) \le x \wedge x \le g^5(x)) \vee (h(b) \le b \wedge b \le h(b)) \}.$$

Perhaps not seen at the first glance, this formula is valid. Assuming the integer numbers as domain, the application of the procedure in figure 4.2 results in 24 (3*2*2*2) conjunctions $\mathcal{G}_i$ in the expansion of the negated formula $\neg\mathcal{F}$ into its disjunctive normal form, where each of the $\mathcal{G}_i$'s conjuncts linear lequalities of the form $\mathcal{A} \le \mathcal{B}$. The well-definedness of functions interpreting the function symbols in the $\mathcal{G}_i$ is violated for at least three associated ILP's for each of which an appropriate formula $\mathcal{H}_j$ has to be constructed. Recalling that in addition each $\mathcal{G}_i$ has to be proven unsatisfiable, at least 27 ILP's have to be solved.
Indeed, there are more than 27 because the formulas $\mathcal{H}_j$ have at least two conjunctions in the disjunctive normal form which has a multiplicative effect on the number of ILP's.
Though the transformation of $\mathcal{A}=\mathcal{B}$ and $\mathcal{A}\neq\mathcal{B}$ to $(\mathcal{A}\le\mathcal{B} \wedge \mathcal{B}\le\mathcal{A})$ and $(\mathcal{A}+1\le\mathcal{B} \vee \mathcal{B}+1\le\mathcal{A})$, respectively, during the expansion advantageously standardizes the structure of the conjunctions $\mathcal{G}_i$, the disadvantage that the given information is not represented in a compact form becomes now evident.
The other way round: Compactly represented information may allow a compact and quick computation. So what about trying to change the direction and reduce $(\mathcal{A}\le\mathcal{B} \wedge \mathcal{B}\le\mathcal{A})$ and $(\mathcal{A}+1\le\mathcal{B} \vee \mathcal{B}+1\le\mathcal{A})$ to $\mathcal{A}=\mathcal{B}$ and $\mathcal{A}\neq\mathcal{B}$, respectively?
Let us reduce the above formula $\mathcal{F}$ to

$$\mathcal{F} \equiv \{ f(a,c)=a \vee g(x)=x \vee (h^3(b)=b \wedge h^5(b)=b) \} \supset \{ f(f(a,c),c)=a \vee g^5(x)=x \vee h(b)=b \},$$

which, using the relations = and ≠, results in only three conjunctions after negating and expanding into disjunctive normal form:

$\mathcal{G}_1 = \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b,\ f(a,c)=a \}$
$\mathcal{G}_2 = \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b,\ g(x)=x \}$
$\mathcal{G}_3 = \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b,\ h^3(b)=b,\ h^5(b)=b \}.$

Aren't we now itching to use the equations as rewrite rules in order to prove straightforwardly each of the G_i's unsatisfiable ?
In this example, we have automatically directed our attention to the new concepts:
Compactifying information by additional usage of the relations = and ≠, and, especially pointed out in the last conjunction, introducing the concept of rewriting.
Before presenting an algorithm, we provide the basic notions of rewriting.

4.4. Rewriting

We now introduce the concepts of rewriting and point out the connection to the decision problem of the quantifier-free Presburger Arithmetic extended by predicate and function symbols.
The notations in section 4.4.1 and the algorithm in section 4.4.2 can be found in the paper "Finding canonical rewriting systems equivalent to a finite set of ground equations in polynomial time" of Gallier, Narendran, Plaisted, Raatz, and Snyder [7].
Sections 4.4.3 and 4.4.4 establish the relation to the class $\mathcal{PA}_f$ and introduce the notion of Presburger terms.

4.4.1. Notations

Let $\Rightarrow \subseteq \mathcal{A} \times \mathcal{A}$ be a binary relation on a set $\mathcal{A}$. The transitive closure, the transitive and reflexive closure, and the inverse of $\Rightarrow$ are denoted by $\Rightarrow^+$, $\Rightarrow^*$, and $\Rightarrow^{-1}$ or $\Leftarrow$, respectively. The relation $\Rightarrow$ is *Noetherian* or *well founded* if and only if there is no infinite sequence $\langle a_0, a_1, \ldots, a_n, \ldots \rangle$ of elements in $\mathcal{A}$ with $a_n \Rightarrow a_{n+1}$ for all $n \geq 0$.
A *partial order* $\preceq$ on a set $\mathcal{A}$ is a binary relation $\preceq \subseteq \mathcal{A} \times \mathcal{A}$ that is reflexive, transitive, and antisymmetric. We let $\succeq = \preceq^{-1}$. Associated with a partial order $\preceq$ on a set $\mathcal{A}$ is a *strict ordering* $\prec$ defined by $a \prec a'$ if and only if $a \preceq a'$ and $a \neq a'$. We let $\succ = \prec^{-1}$. A strict ordering on a set is *well founded* if and only if $\succ$ is well founded according to the above definition.

Let the set $\mathcal{T}$ of ground terms and the set $\mathcal{T}_0$ of terms without loss of generality be constructed by a *ranked* set $\mathcal{U}$ of variables and a *ranked* family $(\mathcal{F}_i)_{i \geq 0}$ of function symbols.

A strict ordering $\prec$ on (ground) terms is *monotonic* if and only if for every two terms s, t and for every function symbol f, if $s \prec t$, then $f(\ldots, s, \ldots) \prec f(\ldots, t, \ldots)$. The strict ordering $\prec$ has the *subterm property* if and only if $t \prec f(\ldots, t, \ldots)$ for every term $f(\ldots, t, \ldots)$.
A *simplification ordering* $\prec$ is a strict ordering that is monotonic and has the subterm property.
Note that if a strict ordering $\prec$ is total, monotonic, and well founded, we must have $s \prec f(\ldots, s, \ldots)$ for every s, since otherwise, by monotonicity, we would have an infinite decreasing chain. From this, we have immediately that for a finite ranked alphabet, a total monotonic ordering $\prec$ is well

founded if and only if it is a simplification ordering. Such an ordering will be called a *total simplification ordering* on ground terms.
It is shown by Dershowitz in [5] that for finite ranked alphabets, any simplification ordering is well founded, and that there exist total simplification orderings on ground terms.
Let $\mathcal{E} \subseteq \mathcal{T} \times \mathcal{T}$ be a binary relation on ground terms. We define the relation $\rightarrow_{\mathcal{E}} \subseteq \mathcal{T} \times \mathcal{T}$ as follows: Given any two terms $t_1, t_2 \in \mathcal{T}$, $t_1 \rightarrow_{\mathcal{E}} t_2$ if and only if there is some pair $(s,t) \in \mathcal{E}$ such that s is a subterm of t_1, and t_2 is obtained by replacing this occurrence of s by t.
When $t_1 \rightarrow_{\mathcal{E}} t_2$, we say that t_1 *rewrites* to t_2, or that we have a *rewrite step*. When a pair (s,t) is used in a rewrite step, we also call it a *rewrite rule* (or *rule*), and use the notation $s \rightarrow t$ to emphasize its use as a rewrite rule. The idea is that the pair is used *oriented* from left to right.
Denote the reflexive and transitive closure, and the symmetric closure of $\rightarrow_{\mathcal{E}}$ by $\rightarrow_{\mathcal{E}}^*$ and $\longleftrightarrow_{\mathcal{E}}$, respectively. Similiarly, the reflexive and transitiv closure of $\longleftrightarrow_{\mathcal{E}}$ is denoted by $\longleftrightarrow_{\mathcal{E}}^*$.

Given a set $\mathcal{R}$ of ground rewrite rules and a total simplification ordering $\prec$, we say that $\mathcal{R}$ is *compatible with* $\prec$ if and only if $r \prec l$ for every rule $l \rightarrow r$ in $\mathcal{R}$.

Given a set $\mathcal{R}$ of ground rewrite rules, we say that $\mathcal{R}$ is *reduced* if and only if neither any lefthand side l of a rewrite rule $l \rightarrow r \in \mathcal{R}$ is reducible by any rewrite rule in $\mathcal{R} - \{l \rightarrow r\}$ nor is any righthand side r of a rewrite rule $l \rightarrow r \in \mathcal{R}$ reducible by any rewrite rule in $\mathcal{R}$.

Let $\rightarrow \subseteq \mathcal{T} \times \mathcal{T}$ be a binary relation on $\mathcal{T}$. We say that $\rightarrow$ is *locally confluent* if and only if for all $t, t_1, t_2 \in \mathcal{T}$, if $t \rightarrow t_1$ and $t \rightarrow t_2$, then there is some t' such that $t_1 \rightarrow^* t'$ and $t_2 \rightarrow^* t'$. We say that $\rightarrow$ is *confluent* if and only if for all $t, t_1, t_2 \in \mathcal{T}$, if $t \rightarrow^* t_1$ and $t \rightarrow^* t_2$, then there is some t' such that $t_1 \rightarrow^* t'$ and $t_2 \rightarrow^* t'$.
It is well known [11] that a Noetherian relation is confluent if and only if it is locally confluent. We say that a set of rewrite rules $\mathcal{R}$ is Noetherian, locally confluent, or confluent if and only if the relation $\rightarrow_{\mathcal{R}}$ associated with $\mathcal{R}$ has the corresponding property.

We say that $\mathcal{R}$ is *canonical* if and only if it is Noetherian and confluent.

Note that since a reduced set of ground rewrite rules has no critical pairs, it is locally confluent. A reduced set $\mathcal{R}$ of ground rewrite rules compatible with $\succ$ is also Noetherian because $r \prec l$ for every rule $l \rightarrow r$, and $\prec$ is a simplification ordering.

Since $\mathcal{R}$ is Noetherian and locally confluent, it is confluent.

4.4.2. Canonical ground rewriting systems

Since total reduction orderings on ground terms exist, ground equations can always be oriented, and thus are Knuth Bendix type completion procedures guaranteed to terminate with an equivalent canonical system on input sets of ground equations. Gallier, Narendran, Plaisted, Raatz, and Snyder have presented an $O(n^3)$ algorithm [7]:
"The basic intuition behind the algorithm is the following. Let $\prec$ be a reduction ordering total on ground terms. Given a finite set $\mathcal{E}$ of ground equations, we run a congruence closure algorithm on $\mathcal{E}$, obtaining its congruence closure in the form of a partition Π. Recall that the equivalence

$\mathcal{R} := \phi$
Π := *associated partition for the congruence closure of* $\mathcal{E}$
WHILE *the partition* Π *has some nontrivial equivalence class* DO
  Let ρ *be the smallest element (wrt* $\prec$*) of the set of terms*
    $C_1 \cup \ldots \cup C_n$
  where $C_1, \ldots, C_n$ *are the nontrivial equivalence classes*
  and let $C_i = \{ \rho, \lambda_1, \ldots, \lambda_k \}$ *be the one containing* ρ.
  Let $\mathcal{L} = \{ \lambda_1 \rightarrow \rho, \ldots, \lambda_k \rightarrow \rho \}$.
  Obtain a canonical system $\mathcal{L}'$ *equivalent to* $\mathcal{L}$ *by simplifying the lefthand sides (the righthand side* ρ *is reduced).*
  Obtain $\mathcal{R}'$ *by simplifying the lefthand sides of rules in* $\mathcal{R}$ *using* $\mathcal{L}'$.
  Let $\mathcal{R} = \mathcal{R}' \cup \mathcal{L}'$
  Simplify all terms in Π *using* $\mathcal{L}'$.
ENDWHILE

Figure 4.4: Computing a canonical rewriting system equivalent to a set $\mathcal{E}$ of ground equations in polynomial time

classes of Π consist of the sets of subterms occurring in $\mathcal{E}$ that are congruent modulo $\mathcal{E}$. Let $C_1, \ldots, C_n$ be the nontrivial equivalence classes[19]

19: Equivalence classes containing only one element are called trivial.

of Π. For each class C_i, we can form a set of rules as follows: let ρ_i be the least element of C_i (w.r.t. $\prec$), let $\mathcal{L}_i$ be the set of all rules of the form $\lambda \longrightarrow \rho_i$, where $\lambda \in C_i$, $\lambda \succ \rho_i$. Now, the union of the sets of rules just constructed is almost the answer. The problem is that the sets $\mathcal{L}_i$ may not be reduced. In order to reduce them, some simplification steps must be performed. But care must be exercised ..... to carry out these simplifications in polynomial time. Roughly speaking, the trick is to choose the classes C_i to form the sets of rules $\mathcal{L}_i$ in an order such that the next class selected is the one containing the least element belonging to non-trivial classes. ..... What happens in this algorithm is that congruence closure is performed only once at the beginning, and that every time a new set of rules $\mathcal{L}_i$ is produced (as the result of picking the right class), $\mathcal{L}_i$ is simplified to a canonical set and it is also used to simplify the current partition and the current set of rules formed so far." [7]
Observe that if a Noetherian set $\mathcal{R}$ of ground rewrite rules is reduced, then it is locally confluent and (since Noetherian) also confluent, and thus canonical.
So the algorithm constructs a canonical set of rules equivalent to the input set of ground equations.

4.4.3. Quantifier-free Presburger theory and rewriting

Since variables can occur in formulas of the Presburger theory, the result of the previous section seems not (directly) applicable to the decision problem of the quantifier-free Presburger Arithmetic extended by predicate and function symbols.
We now explain how the result for ground equations of the previous section can be used for this decision problem:
Recall that for deciding the validity of a quantifier-free formula $\mathcal{F}$, it suffices to provide a decision procedure for the satisfiability of a conjunction of atomic formulas, since $\mathcal{F}$ is valid if and only if each conjunction of atomic formulas in the disjunctive normal form of the negation of $\mathcal{F}$ is unsatisfiable.
Like the formula $\mathcal{F}$, such conjunctions are quantifier-free. For such a quantifier-free conjunction C, obtain a quantifier-free ground conjunction C_a by replacing each variable v occurring in C by a new constant symbol a_v. Since the conjunction C and its existential closure $\exists C$ are equisatisfiable, and since the conjunctions $\exists C$ and C_a are also equisatisfiable by skolem-

ization of $\exists C$, the problem of determining the satisfiability of such a quantifier-free conjunction which may contain variables can thus be reduced to the problem of determining the satisfiability of a quantifier-free ground conjunction.
And therefore, the results of section 4.4.2 can serve for the decision problem of the quantifier-free Presburger Arithmetic extended by predicate and function symbols.
In which way this is done is described in sections 4.5 and 4.6.

4.4.4. Presburger terms

Predicate or function symbols are said to be *uninterpreted* wrt a theory if they do not occur in the axioms of this theory.
The interpreted predicate and function symbols of the Presburger Arithmetic are the predicate symbol $<$ and the function symbols 0, 1, and $+$.

We define the terms of the Presburger Arithmetic $\mathcal{T}_{\mathcal{PA}}$, or Presburger terms, wrt a *ranked* set $\mathcal{V}$ of variables and a *ranked* family $(\mathcal{F}_i)_{i\geq 0}$ of function symbols such that $0, 1 \in \mathcal{F}_0$ and $+ \in \mathcal{F}_2$.

Let $\prec$ be a total simplification ordering on the Presburger ground terms.

Since we want to distinguish between terms with the interpreted function symbol $+$ as outermost function symbol and terms with uninterpreted function symbol as outermost function symbol, we denote terms with top symbol $+$ as *sum-terms* and terms with an uninterpreted top symbol as *$\oint$-terms*.

Let $\mathcal{R}_{\mathcal{PA}}$ be a canonical rewrite system on ground terms for the Presburger Arithmetic extended by predicate and function symbols such that the reduced terms have the form $a_1t_1 + a_2t_2 + \ldots + a_nt_n + a_0$ where the a_i are constants and the t_i are $\oint$-terms such that each $\oint$-term t_i is reduced wrt $\mathcal{R}_{\mathcal{PA}}$ for $1\leq i\leq n$ and $t_i \prec t_{i+1}$ for $1\leq i\leq n-1$.

Intuitively, the rewrite system $\mathcal{R}_{\mathcal{PA}}$ reduces the sum-subterms of the $\oint$-terms and shuffles the resulting $\oint$-terms t_i according to the relation $\prec$ to their position. Sum-terms like $at+bt$ are reduced to $\oint$-terms like $(a+b)t$ so that the t_i's are distinct.

For a set $\mathcal{E}$ of ground equations, a canonical rewrite system $\mathcal{R}$ of ground rewrite rules equivalent to $\mathcal{E}$ can be constructed in polynomial time such that $\mathcal{R}$ is reduced wrt $\mathcal{R}_{\mathcal{PA}}$ by the same algorithm as in section 4.4.2. This can be done by interreducing the terms wrt $\mathcal{R}_{\mathcal{PA}}$ between the given computation steps.

4.5. The main theorem for the extended class

Denote by $\mathcal{L}$, or by $\mathcal{L}_{\leq}$, a set of lequalities[20], by $\mathcal{D}$ a set of inequalities, and by $\mathcal{E}$ a set of equalities each of whose elements can be written in the form $at_1+bt_2 \leq c$, $at_1+bt_2 \neq c$, and $at_1+bt_2=c$, respectively, where the Presburger terms t_1 and t_2 are $\natural$-terms and a, b, and c are constants.

For an equality e, let the associated rewrite rule r_e or $r(e)$ be the rewrite rule $t_1 \rightarrow (-b/a)\, t_2 + c/a$ if $t_1 \succ t_2$, and $t_2 \rightarrow (-a/b)t_1 + c/b$ if $t_2 \succ t_1$ where e is the equality $at_1 + bt_2 = c$.
The set $\mathcal{R}(\mathcal{E})$ of rewrite rules for a set $\mathcal{E}$ of equalities consists of those rewrite rules that are associated with the equalities in $\mathcal{E}$.
Denote by the set $\mathcal{E}(\mathcal{R})$ the set of equations which is obtained by replacing $\rightarrow$ in the rewrite rules of a set $\mathcal{R}$ by =.

A special $\natural$-term t_0 is introduced as zero term. It is assumed that it appears only with coefficient zero, while the other terms require - without loss of generality - nonzero coefficients.

Define the graph $G(\mathcal{L})$ for a set $\mathcal{L}$ of linear lequalities as follows:
For each $\natural$-term occurring in $\mathcal{L}$ give $G(\mathcal{L})$ a vertex labeled with this $\natural$-term and for each lequality in $\mathcal{L}$ give $G(\mathcal{L})$ an edge labeled with this lequality such that this edge connects the corresponding vertices of the occurring $\natural$-terms.

The other notations and definitions like path, triple sequence, admissibility, binary operation * on triples, residue, residue (strict) lequality, loop, simple loop, equality loop, infeasible loop, and closed graph from the previous chapter are now simply transferred to such graphs whose nodes are labeled with $\natural$-terms instead of variables and whose edges are labeled with lequalities which are linear in $\natural$-terms instead of variables.

As suggested in section 4.3, we now combine concepts for the relations $\leq$, =, and $\neq$ for a different treatment of conjunctions.

Denote by *a system* $\mathcal{S}$ a triple $(\mathcal{L}, \mathcal{E}, \mathcal{D})$ with components $\mathcal{L}$, $\mathcal{E}$, and $\mathcal{D}$ where $\mathcal{L}$ is a set of lequalities, $\mathcal{E}$ is a set of equalities, and $\mathcal{D}$ is a set of inequalities.

A system $\mathcal{S}$ that contains the equality $1 = 0$ in the set $\mathcal{E}$ of equalities is called a *degenerated system*.

A system $\mathcal{S}$ is like a partition of a conjunction which splits up its ele-

20: In the second chapter, $\mathcal{L}$ has denoted a set of lequalities which are linear *in variables*, here the lequalities are linear in $\natural$-terms.

ments into three groups.

Obtain for a system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ an *associated system* $\mathcal{S}_o=(\mathcal{L}_o,\mathcal{E}_o,\mathcal{D}_o)$ by replacing each f-term t (outermost, if nested) occurring in $\mathcal{S}$ by a new variable v_t.

In the second chapter, the problem of determining the satisfiability of a set $\mathcal{L}$ of lequalities of the unextended class has been reduced to the problem of determining the satisfiability of a set $\mathcal{L}_C$ of lequalities where the graph $\mathcal{G}(\mathcal{L}_C)$ for $\mathcal{L}_C$ is a closed graph for $\mathcal{L}$.
The notion of a closed graph was used to guarantee the completeness of the LOOP_RESIDUE procedure in figure 2.1.
In the same way, we reduce the problem of determining the satisfiability of a system $\mathcal{S}$ of the extended class to the problem of determining the satisfiability of a closed system $\mathcal{S}_C$ where $\mathcal{S}_C$ is a closure for the system $\mathcal{S}$.
We therefore momentarily define the notion of a closed system as follows:

A system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ is called a *closed system* iff

i) $\mathcal{R}(\mathcal{E})$ is a reduced canonical rewrite system,
ii) $\mathcal{L}$ and $\mathcal{D}$ are normalized wrt $\mathcal{R}(\mathcal{E})\cup\mathcal{R}_{\mathcal{PA}}$,
and iii) $\mathcal{L}$ is a strictly closed set[21].

The notions of a solution, of satisfiability, and of well definedness from section 4.2.2 are naturally extended to systems:
Given a system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$, a solution φ of the set $\mathcal{L}\cup\mathcal{E}\cup\mathcal{D}$ is called a *solution* for the system $\mathcal{S}$.
And a system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ is said to be *satisfiable* if and only if the set $\mathcal{L}\cup\mathcal{E}\cup\mathcal{D}$ is satisfiable.

Observe that a degenerated system is not satisfiable, since $1\neq 0$ is an axiom of the Presburger theory. And note that there is only one degenerated closed system: $\mathcal{S}=(\phi,\{1=0\},\phi)$,

A system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ is said to be *well defined* wrt an interpretation φ if $\varphi(f)$ is a well defined function for any function symbol f occurring in $\mathcal{S}$.

According to the notion of equisatisfiability for sets, two systems $\mathcal{S}_1=(\mathcal{L}_1,\mathcal{E}_1,\mathcal{D}_1)$ and $\mathcal{S}_2=(\mathcal{L}_2,\mathcal{E}_2,\mathcal{D}_2)$ are said to be *equisatisfiable* if and only if the two sets $\mathcal{L}_1\cup\mathcal{E}_1\cup\mathcal{D}_1$ and $\mathcal{L}_2\cup\mathcal{E}_2\cup\mathcal{D}_2$ are equisatisfiable.

The following definition of a closure for a system transfers the problem for deciding the satisfiability from the system to its closure [22]:

21: As defined in chapter two, a set $\mathcal{L}$ of lequalities is strictly closed if the graph $\mathcal{G}(\mathcal{L})$ is closed and contains no equality loop.

22: Since the closure of a graph is not uniquely defined because of the choice of the admissible simple loops *modulo cyclic permutation and reversal*, the closure of a system is not uniquely defined either.

A system $\mathcal{S}^C$ is a *closure* of a system $\mathcal{S}$ if and only if

i) $\mathcal{S}^C$ is a closed system,

and ii) $\mathcal{S}^C$ and $\mathcal{S}$ are equisatisfiable.

In the remaining part of this section, we want to develop a suitable criterion for the satisfiability of a closed system.

This is done in two steps:
First, some results for (strictly closed) sets and closed systems containing no function symbols are presented, and then the main result, a criterion for the satisfiability of a closed system without function symbols, is transferred to closed systems containing function symbols using the notion of an associated system.

The Lemmata 4.1 to 4.3 refer to sets, and strictly closed sets of linear lequalities without function symbols:
In Lemma 4.1 we show that the lequalities $\mathcal{L}_{\mathcal{P}\leq}$ labeling a simple equality loop $\mathcal{P}$ in the graph $\mathcal{G}(\mathcal{L})$ of a set $\mathcal{L}$ of lequalities and the equalities $\mathcal{L}_{\mathcal{P}=}$ are equivalid.
Lemmata 4.2 and 4.3 refer to a strictly closed set $\mathcal{L}$ of linear lequalities without function symbols: The lequalities of such sets can be written as strict lequalities and any inequality containing variables of $\mathcal{L}$ can be added without changing the satisfiability of $\mathcal{L}$.

Theorem 4.4 confirms that any equality that is inferable by a satisfiable closed system $(\mathcal{L},\mathcal{E},\mathcal{D})$ without function symbols is inferable by $\mathcal{E}$.
Supported by Lemmata 4.5 and 4.6, we obtain a criterion for the satisfiability of closed systems without function symbols in Lemma 4.7.

The connection between the satisfiability of a closed system without function symbols and the satisfiability of a closed system with function symbols is established in Lemma 4.8 by proving the equisatisfiability of a closed system and its associated system.
We thus finally obtain a suitable criterion for the satisfiability of closed systems (with function symbols) in Theorem 4.9.

Having done this, it remains to provide an algorithm for computing the closure of a system which is presented in the next section.
Altogether, we thus can determine the satisfiability of a system by computing a closure and determining the satisfiability of this closure.

Lemma 4.1:
For a set $\mathcal{L}$ of linear lequalities and an equality loop $\mathcal{P}$ in the graph $G(\mathcal{L})$,

$\mathcal{L}_{\mathcal{P}\leq}$ and $\mathcal{L}_{\mathcal{P}=}$ are equivalid.

Proof:
Since a solution for $\mathcal{L}_{\mathcal{P}=}$ is a solution for $\mathcal{L}_{\mathcal{P}\leq}$, it suffices to show that a solution for $\mathcal{L}_{\mathcal{P}\leq}$ is a solution for $\mathcal{L}_{\mathcal{P}=}$.
Assume on the contrary that a solution φ for $\mathcal{L}_{\mathcal{P}\leq}$ does not satisfy all equalities of $\mathcal{L}_{\mathcal{P}=}$. W. l. o. g. let $t_1, t_2, \ldots, t_n$ label the vertices of the loop $\mathcal{P}$ and let $a_i t_i + b_i t_{i+1} \leq c_i$, $1 \leq i \leq n-1$, and $a_n t_n + b_n t_1 \leq c_n$ be the lequalities labeling the edges of $\mathcal{P}$ such that

$$\varphi(a_i t_i + b_i t_{i+1}) \leq \varphi(c) \quad \text{for all } 1 \leq i \leq n-1,$$

and $\quad \varphi(a_n t_n + b_n t_1) < \varphi(c_n)$.

Let $\langle a, b, c\rangle = \langle a_1, b_1, c_1\rangle * \langle a_2, b_2, c_2\rangle * \ldots * \langle a_{n-1}, b_{n-1}, c_{n-1}\rangle$.
We have

$$\begin{aligned}\langle a(\mathcal{P}), b(\mathcal{P}), c(\mathcal{P})\rangle &= \langle a, b, c\rangle * \langle a_n, b_n, c_n\rangle, \\ &= sgn(a_n)\langle aa_n, -bb_n, ca_n - c_n b\rangle,\end{aligned}$$

and

$$aa_n - bb_n = 0, \quad ca_n - c_n b = 0,$$

since $\mathcal{P}$ is an equality loop. Furthermore, by Lemma 2.2,

$$\varphi(at_1 + bt_n) \leq \varphi(c).$$

Observing $sgn(b) = sgn(b_{n-1}) = -sgn(a_n)$, we have also

$$\varphi(sgn(a_n)\, a_n (at_1 + bt_n)) \leq \varphi(sgn(a_n) a_n c)$$

and

$$\varphi(-sgn(a_n)\, b (a_n t_n + b_n t_1)) < \varphi(-sgn(a_n) b c_n)$$

implying

$$\varphi(sgn(a_n)(aa_n - bb_n) t_1) < \varphi(sgn(a_n)(ca_n - c_n b)).$$

But since we have $aa_n - bb_n = 0$ and $ca_n - c_n b = 0$, we have $\varphi(0) < \varphi(0)$, a contradiction. We thus have that a solution for $\mathcal{L}_{\mathcal{P}\leq}$ satisfies any equality of $\mathcal{L}_{\mathcal{P}=}$ which completes this proof. □

Lemma 4.2:
For a strictly closed set $\mathcal{L}$ of linear lequalities without function symbols,

$\mathcal{L}_{\leq}$ and $\mathcal{L}_{<}$ are equisatisfiable.

Proof.
We simply have by Theorem 2.3 and its extension to strict lequalities, and by the absence of simple equality loops in the graph $G(\mathcal{L})$ since $\mathcal{L}$ is stric-

tly closed, that

$\mathcal{L}_{\leq}$ is unsatisfiable

iff $\mathcal{G}$ has an admissible simple loop $\mathcal{P}$ with $a(\mathcal{P})+b(\mathcal{P})=0$ and $c(\mathcal{P})<0$

iff $\mathcal{G}$ has an admissible simple loop $\mathcal{P}$ with $a(\mathcal{P})+b(\mathcal{P})=0$ and $c(\mathcal{P})\leq 0$

iff $\mathcal{L}_{<}$ is unsatisfiable □

Lemma 4.3:
For a strictly closed set $\mathcal{L}$ of linear lequalities without function symbols, and for any inequality $l_{\neq}$, say $av+bv' \neq c$, where v and v' are variables occurring in $\mathcal{L}$,

$\mathcal{L}$ and $\mathcal{L} \cup \{ av+bv' \neq c \}$ are equisatisfiable.

Proof:
Since a solution for $\mathcal{L} \cup \{ av+bv' \neq c \}$ is also a solution for $\mathcal{L}$, it suffices to show that $\mathcal{L} \cup \{ av+bv' \neq c \}$ is satisfiable if $\mathcal{L}$ is satisfiable.

Now, if $\mathcal{L}=\mathcal{L}_{\leq}$ is satisfiable, then, by Lemma 4.2, the closedness of $\mathcal{L}$, and the absence of simple equality loops in its graph, $\mathcal{L}_{<}$ is also satisfiable. Let φ be an interpretation that satisfies $\mathcal{L}_{<}$.

We now claim that $\mathcal{L} \cup \{ av+bv' \neq c \}$ is satisfiable.
If $\varphi(av+bv') \neq \varphi(c)$, or, equivalently, $\varphi(c-(av+bv')) \neq 0$, then φ is a solution for $\mathcal{L} \cup \{ av+bv' \neq c \}$ and we are done.
On the other side, if $\varphi(av+bv')= \varphi(c)$, or, equivalently, $\varphi(c-(av+bv'))=0$, we construct an interpretation φ' as follows:
Let $\mathcal{L}_v \subseteq \mathcal{L}$ be that subset of $\mathcal{L}$ that contains all lequalities $\{a_iv+b_iv_i \leq c_i\}$ in which v occurs and let us define by ε the elbowroom for this vertex v:

$$\varepsilon = \textit{minimum}\ (\{\varphi(c_i - a_iv - b_iv_i)/a_i sgn(a_i) \mid (a_iv+b_iv_i \leq c) \in \mathcal{L}_v\}).$$

Since $\mathcal{L}_v$ is a subset of $\mathcal{L}$ and $\mathcal{L}$ is strictly satisfied by φ, we have

$$\varphi(c_i - a_iv - b_iv_i) > 0 \quad \text{and} \quad a_i sgn(a_i) > 0$$

for all $(a_iv+b_iv_i \leq c) \in \mathcal{L}_v$ and thus $\varepsilon > 0$.

We now define the interpretation φ' by

$$\varphi' = \varphi\,[v \longrightarrow (\varphi(v) + \varepsilon/2)].$$

It is momentarily shown that the value for the variable v is changed in such a way that the interpretation additionally satisfies the inequality $l_{\neq}$.
We claim that φ' is a solution for $\mathcal{L}_{<} \cup \{ av+bv' \neq c \}$ and prove this in the following two steps by distinguishing between $\mathcal{L}_{<}$ and $\{ av+bv' \neq c \}$:

i) φ' is a solution for $\mathcal{L}_<$:
Let l be a lequality of $\mathcal{L} - \mathcal{L}_v$. Since l does not contain the variable v, and since φ satisfies $l_<$, φ' also satisfies $l_<$.
Now let l be a lequality of $\mathcal{L}_v$, where l is $a_i v + b_i v_i \leq c$. We have

$$\begin{aligned} & \varphi'(c_i - a_i v - b_i v_i) \\ = \; & \varphi'(c_i) - a_i \varphi'(v) - b_i \varphi'(v_i) \\ = \; & \varphi(c_i) - a_i(\varphi(v) + \varepsilon/2) - b_i \varphi(v_i) \\ = \; & \varphi(c_i - a_i v - b_i v_i) - a_i \varepsilon/2 \end{aligned}$$

Recall that $\varphi(c_i - a_i v - b_i v_i) > 0$ and $\varepsilon > 0$. If $a_i < 0$, then $-a_i\varepsilon/2 > 0$ and $\varphi(c_i - a_i v - b_i v_i) - a_i\varepsilon/2 > 0$. If otherwise $a_i > 0$, i.e., $sgn(a_i) = 1$, we then have by definition of ε that $\varphi(c_i - a_i v - b_i v_i)/a_i sgn(a_i) \geq \varepsilon > \varepsilon/2$ and thus

$$\begin{aligned} & \varphi(c_i - a_i v - b_i v_i) - a_i(\varepsilon/2) \\ > \; & \varphi(c_i - a_i v - b_i v_i) - a_i(\varphi(c_i - a_i v - b_i v_i)/a_i sgn(a_i)) \\ = \; & 0. \end{aligned}$$

So in both cases $a_i > 0$ and $a_i < 0$, we have $\varphi'(c_i - a_i v - b_i v_i) > 0$. Therefore, φ' also satisfies $l_<$ for $l \in \mathcal{L}_v$. Altogether, φ' satisfies $\mathcal{L}_<$.

ii) φ' satisfies $l_\neq$ where $l_\neq$ is $av + bv' \neq c$:
Since $\varphi(c - (av + bv')) = 0$, we have

$$\begin{aligned} & \varphi'(c - (av + bv')) \\ = \; & \varphi'(c) - (a\varphi'(v) + b\varphi'(v')) \\ = \; & \varphi(c) - (a(\varphi(v) + \varepsilon/2) + b\varphi(v')) \\ = \; & \varphi(c - (av + bv')) - a(\varepsilon/2) \\ = \; & 0 - a(\varepsilon/2) \\ \neq \; & 0. \end{aligned}$$

Since this is equivalent to $\varphi'(av + bv') \neq \varphi'(c)$, φ' satisfies $l_\neq$.

By i) and ii), we have proved that φ' is a solution for $\mathcal{L}_< \cup \{ av + bv' \neq c \}$ and thus is $\mathcal{L}_< \cup \{ av + bv' \neq c \}$ satisfiable.
Moreover, this implies that $\mathcal{L} \cup \{ av + bv' \neq c \} = \mathcal{L} \cup \{l_\neq\}$ is satisfiable. □

Theorem 4.4:
i) A nontrivial[23] equality $l_=$, say $av + bv' = c$, can not be inferred by a satisfiable, strictly closed set $\mathcal{L}$ of linear lequalities without function symbols.
ii) If an equality is inferable from a satisfiable closed system $\mathcal{S} = (\mathcal{L}, \mathcal{E}, \mathcal{D})$ without function symbols, then it is inferable from the equality set $\mathcal{E}$.

23: An equality $av + bv' = c$ is trivial if it can be reduced by $\mathcal{R}_{\mathcal{DA}}$ to the equality $0 = 0$.

Proof:
i) Let an interpretation φ be a solution for the satisfiable set $\mathcal{L}$, and obtain the interpretations φ_v and $\varphi_{v'}$ by $\varphi[v \rightarrow (\varphi(v)+1)]$ and $\varphi[v' \rightarrow (\varphi(v')+1)]$, respectively.
If either v or v' does not label a vertex of the graph $G(\mathcal{L})$, then one of the interpretations (φ or φ_v) or (φ or $\varphi_{v'}$), respectively, satisfies $\mathcal{L} \cup \{l_{\neq}\}$.
If both terms v and v' label vertices of the graph $G(\mathcal{L})$, the set $\mathcal{L} \cup \{l_{\neq}\}$ is satisfiable by Lemma 4.3.
Altogether, we have that $l_{=}$ can not be inferred from $\mathcal{L}$.
ii) Obviously, since by i) no equality can be inferred by $\mathcal{L}$, and no equality can be inferred by $\mathcal{D}$. □

Lemma 4.5:
For a closed system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ without function symbols where $\mathcal{D}$ does not contain the inequality $0 \neq 0$,

$$\mathcal{L} \text{ and } \mathcal{L} \cup \mathcal{D} \text{ are equisatisfiable.}$$

Proof:
Since a solution for $\mathcal{L} \cup \mathcal{D}$ is also a solution for $\mathcal{L}$, it suffices to show that $\mathcal{L} \cup \mathcal{D}$ is satisfiable if $\mathcal{L}$ is satisfiable.

Let $\mathcal{D} = \{ a_i v_i + b_i v_i' \neq c_i \mid 1 \leq i \leq n \}$
and $\mathcal{D}_j = \{ a_i v_i + b_i v_i' \neq c_i \mid 1 \leq i \leq j \}$ for $0 \leq j \leq n$.
We now inductively construct a sequence $\langle \mathcal{L}_0, \mathcal{L}_1, \ldots \mathcal{L}_n \rangle$ of sets of lequalities such that for each $0 \leq i \leq n$, the following conditions i) and ii) are satisfied:

i) $\mathcal{L}_{i<}$ is satisfiable, and
ii) $\mathcal{L}_i$ and $\mathcal{L} \cup \mathcal{D}_i$ are equisatisfiable.

Basis:
Let $\mathcal{L}_0 = \mathcal{L}$. $\mathcal{L}_{0<}$ is satisfiable by the satisfiability of $\mathcal{L}$ and Lemma 4.2. And since $\mathcal{D}_0 = \emptyset$, $\mathcal{L}_0$ and $\mathcal{L} \cup \mathcal{D}_0$ are equisatisfiable.

Induction Step: ($i<n$)
Since $\mathcal{L}_{i<}$ is satisfiable, there is a solution φ for $\mathcal{L}_{i<}$.
It can in a similiar way as in Theorem 4.4 be proved that $\mathcal{L}_{i<} \cup \{ a_{i+1} v_{i+1} + b_{i+1} v_{i+1}' \neq c_{i+1} \}$ is satisfiable by constructing an appropriate interpretation φ' in case of $\varphi(a_{i+1} v_{i+1} + b_{i+1} v_{i+1}') = \varphi(c_{i+1})$ such that $\varphi'(a_{i+1} v_{i+1} + b_{i+1} v_{i+1}') \neq \varphi'(c_{i+1})$.
Recall that the equality $a_{i+1} v_{i+1} + b_{i+1} v_{i+1}' = c_{i+1}$ must be nontrivial because of the closedness of the system $\mathcal{S}$.

If now $\varphi'(a_{i+1}v_{i+1}+b_{i+1}v_{i+1}') < \varphi'(c_{i+1})$,
then let $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{ a_{i+1}v_{i+1}+b_{i+1}v_{i+1}' \leq c_{i+1} \}$.
If otherwise $\varphi'(a_{i+1}v_{i+1}+b_{i+1}v_{i+1}') > \varphi'(c_{i+1})$,
then let $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{ -a_{i+1}v_{i+1}-b_{i+1}v_{i+1}' \leq -c_{i+1} \}$.

We have that both $\mathcal{L}_{i+1<}$ and $\mathcal{L} \cup \mathcal{D}_{i+1}$ are satisfiable by the same interpretation φ'. Therefore both conditions i) and ii) hold.
So finally, we have that $\mathcal{L}_{n<}$ is satisfiable and $\mathcal{L}_n$ and $\mathcal{L} \cup \mathcal{D}_n$ are equisatisfiable. Thus is $\mathcal{L} \cup \mathcal{D}$ satisfiable.
We needed the sequence $<\mathcal{L}_0, \mathcal{L}_1, \ldots \mathcal{L}_n>$ of sets of lequalities in order to refer the construction of a solution in the induction step to that of Theorem 4.4. □

Lemma 4.6:
Given a non degenerated closed system $\mathcal{S}_\mathcal{E}=(\mathcal{L},\mathcal{E},\mathcal{D})$ without function symbols, then the systems

$$\mathcal{S}=(\mathcal{L},\emptyset,\mathcal{D}) \text{ and } \mathcal{S}_\mathcal{E} \text{ are equisatisfiable.}$$

Proof:
Since a solution for the system $\mathcal{S}_\mathcal{E}$ is also a solution for the system $\mathcal{S}$, it remains to prove the reverse direction that $\mathcal{S}_\mathcal{E}$ is satisfiable if $\mathcal{S}$ is satisfiable.

So let φ be a solution for $\mathcal{S}$.
We inductively construct a finite sequence $<\varphi_0, \varphi_1, \ldots, \varphi_n>$ of interpretations where n is less or equal to the number of equalities in $\mathcal{E}$ such that for every i, $0 \leq i \leq n$,

i) φ_i is a solution for $\mathcal{S}$, and
ii) φ_i satisfies at least i equalities of $\mathcal{E}$.

Basis:
Let $\varphi_0 = \varphi$. Both conditions i) and ii) hold since φ is a solution for $\mathcal{S}$ and $i=0$.

Induction Step:
From the induction hypothesis, φ_i is a solution for $\mathcal{S}$. If it is also a solution for $\mathcal{S}_\mathcal{E}$, we are done.
So if φ_i does not satisfy the system $\mathcal{S}_\mathcal{E}$, there is an equality $av+bv'=c$ in $\mathcal{E}$ such that $\varphi_i(av+bv') \neq \varphi_i(c)$. Let w.l.o.g. $v' \rightarrow (-a/b)v + c/b$ be the associated rewrite rule for this equality.[24]

24: The system $\mathcal{S}$ can not contain a nontrivial equality of the form $c=c'$ where c and c' are constants, since by its closedness $\mathcal{S}$ would then contain the equality $1=0$ in contradiction to the assumption that $\mathcal{S}$ is not degenerated.
The unsatisfied equality is thus of either form of $bv'=c$ or $av+bv'=c$. By adding $0v_0$ in the first case, we have $av+bv'=c$ in both cases.

Since $\mathcal{S}_{\mathcal{E}}$ is a closed system, the sets $\mathcal{L}$ and $\mathcal{D}$ are reduced wrt the equations in $\mathcal{E}$, and therefore does the variable v' not occur in $\mathcal{L}$, $\mathcal{D}$, and $\mathcal{E}-\{ av+bv'=c \}$.
Let φ_{i+1} be the interpretation defined by

$$\varphi_{i+1} = \varphi_i [v' \rightarrow (\varphi_i((-a/b)v + c/b))].$$

The interpretation φ_{i+1} is a solution for $\mathcal{S}$. It satisfies the equations of $\mathcal{E}$ that are satisfied by φ_i, and, since

$$\begin{aligned} \varphi_{i+1}(av+bv') &= a\varphi_{i+1}(v) + b\varphi_{i+1}(v') \\ &= a\varphi_i(v) + b\ \varphi_i((-a/b)v + c/b) \\ &= a\varphi_i(v) + b(-a/b)\ \varphi_i(v) + b\varphi_i(c/b) \\ &= \varphi_i(c) = \varphi_{i+1}(c), \end{aligned}$$

it satisfies at least one more equality of $\mathcal{E}$ than φ_i. And since φ_i satisfies at least i equalities of $\mathcal{E}$, φ_{i+1} satisfies at least $i+1$ equalities. So both i) and ii) hold for φ_{i+1}.

Since $\mathcal{E}$ is finite, there exists an n such that φ_n is a solution for $\mathcal{S}$ and φ_n satisfies all equalities of $\mathcal{E}$. So the system $\mathcal{S}_{\mathcal{E}}$ is satisfiable. □

Lemma 4.7:
A closed system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ without function symbols is unsatisfiable if and only if either

i) $\mathcal{L}$ is unsatisfiable,
or ii) $1=0 \in \mathcal{E}$,
or iii) $0\neq 0 \in \mathcal{D}$.

Proof:
If $\mathcal{S}$ is satisfiable, then both $\mathcal{L}$ must be satisfiable, and neither $\mathcal{E}$ nor $\mathcal{D}$ can contain the equality $1=0$ or the inequality $0\neq 0$, respectively.

Now conversely, if both $\mathcal{L}$ is satisfiable and $\mathcal{D}$ does not contain the inequality $0\neq 0$, then, by Lemma 4.5 and the closedness of $\mathcal{S}$, $\mathcal{L} \cup \mathcal{D}$ is satisfiable.
Since $(\mathcal{L},\phi,\mathcal{D})$ is satisfiable and since $\mathcal{E}$ does not contain the equality $1=0$, $\mathcal{S}$ is a non degenerated closed system and we have by Lemma 4.6 that $(\mathcal{L},\mathcal{E},\mathcal{D}) = \mathcal{S}$ is satisfiable. □

Lemma 4.7 thus provides a criterion for the satisfiability of an associated system $\mathcal{S}_v$ for a closed system $\mathcal{S}$.
By the main theorem for the extended class following the next lemma, the same criterion is applicable for a closed system $\mathcal{S}$ itself.

Lemma 4.8:
If the system $\mathcal{S}_\upsilon$ is an associated system for the closed system $\mathcal{S}$,

$\mathcal{S}_\upsilon$ and $\mathcal{S}$ are equisatisfiable.

Proof:
Let C_υ and C be the conjunctions corresponding to the systems $\mathcal{S}_\upsilon$ and $\mathcal{S}$, and let the formulas $\mathcal{F}_\upsilon$ and $\mathcal{F}_\mathfrak{f}$ be $\neg C_\upsilon$ and $\neg C$, respectively.
In correspondence to the reduction process of the decidability proof in section 4.1, let the formula $\mathcal{F}_{\mathcal{A}\mathfrak{f}}$ be obtained by

$$\mathcal{F}_{\mathcal{A}\mathfrak{f}} \equiv (\mathcal{A}_{\mathfrak{f}1} \wedge \mathcal{A}_{\mathfrak{f}2} \wedge \ldots \wedge \mathcal{A}_{\mathfrak{f}t}) \supset \mathcal{F}_\mathfrak{f}$$

where $\mathcal{F}_{\mathcal{A}\mathfrak{f}}$ contains an axiom $\mathcal{A}_\mathfrak{f}$ for each pair $\mathfrak{f}(t_1, \ldots, t_n)$, $\mathfrak{f}(u_1, \ldots, u_n)$ of distinct $\mathfrak{f}$-terms with the same outermost n-ary function symbol $\mathfrak{f}$. Recall that the axioms are of the form

$$t_1=u_1 \wedge t_2=u_2 \wedge \ldots \wedge t_n=u_n \supset \mathfrak{f}(t_1, \ldots, t_n)=\mathfrak{f}(u_1, \ldots, u_n).$$

The formula $\mathcal{F}_{\mathcal{A}\upsilon}$ is obtained by replacing each (outermost) $\mathfrak{f}$-term t occurring in $\mathcal{F}_{\mathcal{A}\mathfrak{f}}$ by a new variable υ_t. The formula $\mathcal{F}_{\mathcal{A}\upsilon}$ now contains axioms of the form

$$\upsilon_1=\upsilon_1' \wedge \upsilon_2=\upsilon_2' \wedge \ldots \wedge \upsilon_n=\upsilon_n' \supset \upsilon_\mathfrak{f} = \upsilon_\mathfrak{f}'.$$

We now have that

$\mathcal{S}_\upsilon$ is unsatisfiable
iff C_υ is unsatisfiable
iff $\mathcal{F}_\upsilon$ is valid

and

$\mathcal{F}_{\mathcal{A}\upsilon}$ is valid
iff $\mathcal{F}_{\mathcal{A}\mathfrak{f}}$ is valid
iff $\mathcal{F}_\mathfrak{f}$ is valid
iff C is unsatisfiable
iff $\mathcal{S}$ is unsatisfiable

by the result of section 4.1.
Using these equivalences and the closedness of the systems $\mathcal{S}_\upsilon$ and $\mathcal{S}$, we now prove the equisatisfiability of $\mathcal{S}_\upsilon$ and $\mathcal{S}$.
It again suffices to show that $\mathcal{S}$ is satisfiable if $\mathcal{S}_\upsilon$ is satisfiable, since the other direction holds obviously. We thus assume that $\mathcal{S}_\upsilon$ is satisfiable and show that $\mathcal{S}$ must also be satisfiable:
Inductively construct a sequence $\langle \mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_t \rangle$ of systems such that $\mathcal{L}_i=\mathcal{L}_\upsilon$, $\mathcal{E}_i=\mathcal{E}_\upsilon$ and the following conditions hold:

i) S_i is a satisfiable closed system without function symbols
ii) a solution for S_i also satisfies $A_{v1} \wedge A_{v2} \wedge \ldots \wedge A_{vi} \wedge C_v$.

Basis:
Let $S_0=S_v$; S_0 is satisfiable, and S_0 and C_v are equivalid: Both conditions i) and ii) are satisfied.

Induction Step:
Let $0 \le i \le t-1$.
By the induction hypothesis, we have that S_i is a satisfiable closed system without function symbols. Let axiom $A_{v,i+1}$ and the corresponding axiom $A_{f,i+1}$ of the formula $\mathcal{F}_{Af}$ be

$$v_1=v_1' \wedge v_2=v_2' \wedge \ldots \wedge v_n=v_n' \supset v_f = v_f'$$

and

$$t_1=u_1 \wedge t_2=u_2 \wedge \ldots \wedge t_n=u_n \supset f(t_1, \ldots, t_n)=f(u_1, \ldots, u_n).$$

CASE 1:
There is an interpretation φ that satisfies $S_i \cup \{v_j \neq v_j'\}$ for some j, $1 \le j \le n$.
Let S_{i+1} be obtained from S_i by reducing the inequality $v_j \neq v_j'$ wrt the set $\mathcal{R}(\mathcal{E}_i)=\mathcal{R}(\mathcal{E}_v)$ of rewrite rules and augmenting the set $\mathcal{D}_i$ by this inequality.
Obviously, condition i) is satisfied.
And since a solution φ for S_{i+1} is also a solution for S_i, we have by the induction hypothesis that the formula $A_{v1} \wedge A_{v2} \wedge \ldots \wedge A_{vi} \wedge C_v$ is also satisfied by φ. Moreover, since φ is a solution for S_{i+1}, it also satisfies that inequality which is obtained by reducing $v_j \neq v_j'$ wrt the set $\mathcal{R}(\mathcal{E}_v)$. Therefore, the interpretation φ also satisfies the axiom $A_{v,i+1}$. Condition ii) thus holds for this choice of S_{i+1}.

CASE 2:
Any interpretation that satisfies S_i also satisfies all equalities of the set $\{v_1=v_1', v_2=v_2', \ldots, v_n=v_n'\}$.
By the result of Theorem 4.4 for a satisfiable closed system without function symbols, all equalities of the set $\{v_1=v_1', v_2=v_2', \ldots, v_n=v_n'\}$ are inferable from the set $\mathcal{E}_i=\mathcal{E}_v$ of equalities.
By the construction of the associated system S_v, the equalities $t_1=u_1$, $t_2=u_2$, ..., $t_n=u_n$ must be inferable by the set $\mathcal{E}$. Therefore, the equality $f(t_1, \ldots, t_n)=f(u_1, \ldots, u_n)$ is also inferable by $\mathcal{E}$. Again, by the construction of the associated system S_v, the equality $v_f=v_f'$ is inferable by $\mathcal{E}_v$.

We thus have that any interpretation that satisfies $\mathcal{S}_i$ also satisfies $A_{0,i+1}$. We simply let $\mathcal{D}_{i+1}=\mathcal{D}_i$. Thus both conditions i) and ii) hold for $\mathcal{S}_{i+1}=\mathcal{S}_i$.
We have thus proved that the formula

$$A_{01} \wedge A_{02} \wedge \ldots \wedge A_{0t} \wedge C_0$$

is satisfiable. Recalling $\mathcal{F}_0 \equiv \neg C_0$, the formula

$$\mathcal{F}_{A0} \equiv (A_{01} \wedge A_{02} \wedge \ldots \wedge A_{0t}) \supset \mathcal{F}_0$$

is therefore not valid. As a result by the equivalence chain

$\mathcal{F}_{A0}$ is valid
iff $\mathcal{F}_{A\beta}$ is valid
iff $\mathcal{F}_\beta$ is valid
iff C is unsatisfiable
iff $\mathcal{S}$ is unsatisfiable,

we have that the system $\mathcal{S}$ is satisfiable. This now completes the proof of this Lemma. □

Theorem 4.9:
A closed system $\mathcal{S}=(\mathcal{L}, \mathcal{E}, \mathcal{D})$ is unsatisfiable if and only if either

i) $\mathcal{L}$ is unsatisfiable,
ii) $1=0 \in \mathcal{E}$,
or iii) $0\neq 0 \in \mathcal{D}$.

Proof.
By Lemmata 4.7 and 4.8, and the construction of an associated system. □

Having this criterion for the satisfiability of a closed system, the remaining task is the computation of a closure for a system. This is done in the next section. We then can determine the satisfiability of a system $\mathcal{S}$ by computing a closure and decide the satisfiability of the closure by the criterion of Theorem 4.9.

4.6. The Equality Loop Residue Procedure

The task of this chapter is to compute a closure $\mathcal{S}_C$ for a system $\mathcal{S}$. We do this by constructing a finite sequence $\mathcal{S}=\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n=\mathcal{S}_C$ of systems where for $1\le i\le n$ the systems $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ are equisatisfiable. Even more, this is done in such a way that they are equivalid.

Define the set $\mathcal{E}_\mathcal{P}$ of equalities for an equality loop $\mathcal{P}$ labeled with $a_1t_1+b_1t_2\le c_1,\ a_2t_2+b_2t_3\le c_2,\ \ldots,\ a_nt_n+b_nt_1\le c_n$ and - without loss of generality - smallest term t_1 (i.e., $t_1 \prec t_i$ for $2\le i\le n$) by

$$\mathcal{E}_\mathcal{P} = \{a_{1i}t_1+b_{1i}t_{i+1}=c_{1i} \mid 1\le i\le n-1\}$$

where $\langle a_{11},b_{11},c_{11}\rangle = \langle a_1,b_1,c_1\rangle$
and $\langle a_{1i},b_{1i},c_{1i}\rangle = \langle a_1,b_1,c_1\rangle * \ldots * \langle a_i,b_i,c_i\rangle$.

Lemma 4.10:
If $\mathcal{P}$ is an equality loop, then $\mathcal{L}_{\mathcal{P}_=}$ and $\mathcal{E}_\mathcal{P}$ are equivalid.

Proof.
Since the triples $\langle a_{1i},b_{1i},c_{1i}\rangle$ are residues of (admissible) subpaths of $\mathcal{P}$, the sets $\mathcal{L}_{\mathcal{P}_=}$ and $\mathcal{L}_{\mathcal{P}_=} \cup \mathcal{E}_\mathcal{P}$ are equivalid by Lemma 2.2.

CLAIM 1:
The sets $\mathcal{L}_{\mathcal{P}_=} \cup \mathcal{E}_\mathcal{P}$ and $\mathcal{E}_\mathcal{P} \cup \{a_nt_n+b_nt_1=c_n\}$ are equivalid.

Proof:
Let $\mathcal{E}_j = (\mathcal{L}_{\mathcal{P}_=} - \{a_it_i+b_it_{i+1}=c_i \mid 1\le i\le j\}) \cup \mathcal{E}_\mathcal{P}$ for $0\le j\le n-1$.
We prove by finite induction on j, $1\le j\le n-1$, that the sets $\mathcal{E}_{j-1}$ and $\mathcal{E}_j$ are equivalid. Since $\mathcal{E}_j \subseteq \mathcal{E}_{j-1}$ and $(\mathcal{E}_{j-1} - \mathcal{E}_j) \subseteq \{a_jt_j+b_jt_{j+1} = c_j\}$, it therefore suffices to show that the equality $a_jt_j+b_jt_{j+1} = c_j$ is inferable by $\mathcal{E}_j$. Observe that by their definition, $\mathcal{E}_\mathcal{P} \subseteq \mathcal{E}_j$ for all $0\le j\le n-1$.

Basis.
Since $a_1t_1+b_1t_2=c_1 \in \mathcal{E}_1$, $\mathcal{E}_0=\mathcal{E}_1$ and thus they are equivalid.

Induction Step.
Now, since $j>1$,

$$a_{1,j-1}t_1+b_{1,j-1}t_j = c_{1,j-1} \in \mathcal{E}_\mathcal{P} \subseteq \mathcal{E}_j ,$$
$$a_{1,j}t_1+b_{1,j}t_{j+1} = c_{1,j} \in \mathcal{E}_\mathcal{P} \subseteq \mathcal{E}_j ,$$

and $\langle a_{1,j},b_{1,j},c_{1,j}\rangle = \langle a_{1,j-1},b_{1,j-1},c_{1,j-1}\rangle * \langle a_j,b_j,c_j\rangle$
$= sgn(a_j)\ \langle a_{1,j-1}a_j,\ -b_{1,j-1}b_j,\ c_{1,j-1}a_j - c_jb_{1,j-1}\rangle$,

we have that the equalities

$$a_{1,j-1}a_jt_1 + b_{1,j-1}a_jt_j = c_{1,j-1}a_j,$$
$$a_{1,j-1}a_jt_1 - b_{1,j-1}b_jt_{j+1} = c_{1,j-1}a_j - c_jb_{1,j-1},$$

and

$$\begin{aligned} & a_i t_i + b_i t_{i+1} \\ = \; & b_{1,i-1} a_i t_i \;/\; b_{1,i-1} \;+\; b_{1,i-1} b_i t_{i+1} \;/\; b_{1,i-1} \\ = \; & (c_{1,i-1} a_i - a_{1,i-1} a_i t_1) \;/\; b_{1,i-1} \\ & + (a_{1,i-1} a_i t_1 - (c_{1,i-1} a_i - c_i b_{1,i-1})) \;/\; b_{1,i-1} \\ = \; & c_i \end{aligned}$$

are inferable by $\mathcal{E}_i$. Therefore, the sets $\mathcal{E}_{i-1}$ and $\mathcal{E}_i$ are equivalid which completes the induction.

Since $\mathcal{E}_0 = \mathcal{L}_{\mathcal{P}=} \cup \mathcal{E}_{\mathcal{P}}$ and $\mathcal{E}_{n-1} = \mathcal{E}_{\mathcal{P}} \cup \{ a_n t_n + b_n t_1 = c_n \}$, the sets $\mathcal{L}_{\mathcal{P}=} \cup \mathcal{E}_{\mathcal{P}}$ and $\mathcal{E}_{\mathcal{P}} \cup \{ a_n t_n + b_n t_1 = c_n \}$ are equivalid.

CLAIM 2:

$\mathcal{E}_{\mathcal{P}} \cup \{ a_n t_n + b_n t_1 = c_n \}$ and $\mathcal{E}_{\mathcal{P}}$ are equivalid.

Proof:

Recalling that

$$a_{1,n-1} t_1 + b_{1,n-1} t_n = c_{1,n-1} \in \mathcal{E}_{\mathcal{P}},$$

and $a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) = 0$,

where

$$\langle a(\mathcal{P}), b(\mathcal{P}), c(\mathcal{P}) \rangle = \langle a_{1,n-1}, b_{1,n-1}, c_{1,n-1} \rangle * \langle a_n, b_n, c_n \rangle$$

we have

$$a_{1,n-1} a_n = b_{1,n-1} b_n \quad \text{and} \quad c_{1,n-1} a_n = c_n b_{1,n-1}$$

Thus,

$$\begin{aligned} a_n t_n + b_n t_1 &= (a_{1,n-1} a_n t_n + a_{1,n-1} b_n t_1) \;/\; a_{1,n-1} \\ &= (b_n b_{1,n-1} t_n + b_n a_{1,n-1} t_1) \;/\; a_{1,n-1} \\ &= b_n c_{1,n-1} \;/\; a_{1,n-1} \\ &= a_n c_{1,n-1} \;/\; b_{1,n-1} \\ &= b_{1,n-1} c_n \;/\; b_{1,n-1} \\ &= c_n. \end{aligned}$$

The equality $a_n t_n + b_n t_1 = c_n$ is thus inferable by $\mathcal{E}_{\mathcal{P}}$ and therefore are $\mathcal{E}_{\mathcal{P}} \cup \{ a_n t_n + b_n t_1 = c_n \}$ and $\mathcal{E}_{\mathcal{P}}$ are equivalid which completes the proof of this claim.

So we have by Lemma 2.2, Claim 1, and Claim 2,

$\mathcal{L}_{\mathcal{P}=}$ is valid
iff $\mathcal{L}_{\mathcal{P}=} \cup \mathcal{E}_{\mathcal{P}}$ is valid
iff $\mathcal{E}_{\mathcal{P}} \cup \{ a_n t_n + b_n t_1 = c_n \}$ is valid
iff $\mathcal{E}_{\mathcal{P}}$ is valid

□

Motivated by the result of Lemma 4.10, we now define the system $S(P)$:
For a system $S=(L,E,O)$ and an equality loop P, denote by the system $S(P)$ the system $(L(P), E \cup E_P, O(P))$, where $L(P)$ and $O(P)$ are obtained by reducing the sets $L-L_P$ and O wrt the rewrite system $R(E_P) \cup R_{PA}$.
Observe that the lefthand sides of the rules in $R(E_P)$ do not occur in $L(P)$ nor in $O(P)$ and that only those lequalities are reduced which incident with a vertex of the equality loop.

Theorem 4.11:
For a system $S=(L,E,O)$ and a simple equality loop P in the graph $G(L)$,

i) S and $S(P)$ are equivalid
ii) the graph $G(L(P))$ has less vertices than the graph $G(L)$

Proof:
i) $S = (L,E,O)$ is valid
iff (by Lemma 4.1)
$(L-L_P, E \cup L_{P=}, O)$ is valid
iff (by Lemma 4.10)
$(L-L_P, E \cup E_P, O)$ is valid
iff (by elimation of $t_2, t_3, \ldots, t_n$ by application of the rules $R(E_P)$)
$S(P) = (L(P), E \cup E_P, O(P))$ is valid
ii) Obviously, since a loop has at least length 2. □

Theorem 4.12:
a) Let $S=(L,E,O)$ be a system and R be a canonical rewrite system such that $E=E(R)$.
Recall that $E(R)$ is obtained from R by replacing each $\rightarrow$ by $=$.
Let $L(R)$ and $O(R)$ be obtained by reducing L and O by the rules in R.
Then are equivalent:

i)	$S=(L,E,O)$	is valid
ii)	$(L,E(R),O)$	is valid
iii)	$(L(R),E(R),O(R))$	is valid.

b) Let S' be obtained from S by reduction wrt R_{PA}, the canonical rewrite system for the Presburger terms.
Then S and S' are equivalid systems of the Presburger theory.

Proof:
Obviously. □

It is assumed that a formula $\mathcal{F}$ of the quantifier-free Presburger Arithmetic extended by predicate and function symbols that is to be examined for validity is equivalently transformed to a formula without predicate symbols (this can be done by the introduction of a new function symbol f_P for each predicate symbol P) which then is negated and expanded into disjunctive normal form so that $\neg\mathcal{F} \equiv S_1 \vee \ldots \vee S_p$ where the S_i's are conjunctions of lequalities, equalities, and inequalities so that they represent the systems $S_i = (\mathcal{L}_i, \mathcal{E}_i, \mathcal{D}_i)$, $1 \le i \le p$. The formula $\mathcal{F}$ is thus valid if and only if each system S_i is unsatisfiable.

The EQUALITY_LOOP_RESIDUE procedure in figure 4.5 can then be used to determine the satisfiability of a system S.

This procedure calls some subprocedures which are now described:

The procedure LOOP_RESIDUE performs a generalized version of the procedure described in the second chapter. It is generalized in the sense that the vertices are labeled by terms and the lequalities are linear in terms.

The procedure LOOP_COLLABATION takes as arguments a system $S = (\mathcal{L}, \mathcal{E}, \mathcal{D})$ and an equality loop P of the graph $G(\mathcal{L})$.
The procedure computes the system $S(P) = (\mathcal{L}(P), \mathcal{E} \cup \mathcal{E}_P, \mathcal{D}(P))$ where $\mathcal{L}(P)$ and $\mathcal{D}(P)$ are obtained by reducing the sets $\mathcal{L} - \mathcal{L}_P$ and $\mathcal{D}$ wrt the rewrite system $\mathcal{R}(\mathcal{E}_P) \cup \mathcal{R}_{\mathcal{PA}}$.
Note that in the FOR-loop where the procedure LOOP_COLLABATION is called, the lequalities of the simple equality loops initially found in the graph may be reduced in the collabation process if the vertices of those loops are not distinct.[25]

The procedure CANONIZATION computes for the set $\mathcal{E}$ of ground equations a canonical rewriting system $\mathcal{R}$ equivalent to $\mathcal{E}$ which is reduced wrt $\mathcal{R}_{\mathcal{PA}}$. According to section 4.4.2, this can be done in polynomial time. Recall that $\mathcal{R}_{\mathcal{PA}}$ is a canonical rewrite system for the Presburger terms.

The procedure REDUCTION takes as arguments a system $S = (\mathcal{L}, \mathcal{E}, \mathcal{D})$ and a canonical set $\mathcal{R}$ of rewrite rules. It produces a system $(\mathcal{L}(\mathcal{R}), \mathcal{E}, \mathcal{D}(\mathcal{R}))$, where $\mathcal{L}(\mathcal{R})$ and $\mathcal{D}(\mathcal{R})$ are reduced wrt $\mathcal{R} \cup \mathcal{R}_{\mathcal{PA}}$.

Theorem 4.13:
The EQUALITY LOOP RESIDUE PROCEDURE terminates.

Proof:
It suffices to show that both REPEAT loops terminate:
In each iteration except the last one, the innermost REPEAT loop reduces the number of vertices in the graph $G(\mathcal{L})$, since by Theorem 4.11 the graph

25: Look at the examples in section 4.8, especially example 6.

```
REPEAT
  RESET (collapsed)
  REPEAT
    SET (continue)
    LOOP_RESIDUE (G(L))
    IF G(L) has an infeasible simple loop
      THEN RETURN (S is unsatisfiable)
      ELSE IF G(L) has no simple equality loop
             THEN RESET (continue)
             ELSE
      BEGIN
        SET (collapsed)
        FOR all simple equality loops P in G(L) DO
        S := LOOP_COLLABATION (S, P)
        IF 1=0 ∈ E OR 0≠0 ∈ D
          THEN RETURN (S is unsatisfiable)
      END
  UNTIL NOT continue
  IF collapsed
    THEN BEGIN
           S := REDUCTION (S, CANONIZATION(E))
           IF 1=0 ∈ E OR 0≠0 ∈ D
             THEN RETURN (S is unsatisfiable)
         END
UNTIL NOT collapsed
S := REDUCTION (S, CANONIZATION(E))
IF 1=0 ∈ E OR 0≠0 ∈ D
  THEN RETURN (S is unsatisfiable)
  ELSE RETURN (S is satisfiable)
```

Figure 4.5: EQUALITY LOOP RESIDUE PROCEDURE deciding the satisfiability of a system S

$G(L(P))$ has less vertices than the graph $G(L)$.
And in each iteration of the outermost REPEAT loop except the last one, at least one equality loop has been replaced by a single vertex in the inner loop. Since the LOOP_RESIDUE procedure terminates and since a graph has finitely many vertices, both REPEAT loops of the procedure terminate. □

Theorem 4.14:
The EQUALITY LOOP RESIDUE PROCEDURE is sound.
Proof:
We have to show that if the procedure answers that a system is unsatisfiable, the answer is correct.
The procedure constructs a sequence of systems. In order to prove the soundness of the procedure, it suffices to prove that every two subsequent systems $\mathcal{S}$ and $\mathcal{S}'$ in the sequence are equisatisfiable. We are even able to show that they are equivalid.[26]
Any modification of the system is performed by a call of either procedure of

i) LOOP_RESIDUE,
ii) LOOP_COLLABATION, or
iii) REDUCTION,

We thus have the following cases for $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ and $\mathcal{S}'=(\mathcal{L}',\mathcal{E}',\mathcal{D}')$:

i) $\mathcal{L}=\mathcal{L}'$, $\mathcal{E}=\mathcal{E}'$, $\mathcal{G}(\mathcal{L}')$ is a closed graph for $\mathcal{G}(\mathcal{L})$.
   Since $\mathcal{L}$ is a subset of $\mathcal{L}'$ and $\mathcal{L}'$ is augmented exclusively by loop residue lequalities of admissible simple loops in the graph $\mathcal{G}(\mathcal{L})$, $\mathcal{L}$ and $\mathcal{L}'$ are equivalid by Lemma 2.2. Therefore, also $\mathcal{S}$ and $\mathcal{S}'$ are equivalid.
ii) $\mathcal{L}'=\mathcal{L}(\mathcal{P})$, $\mathcal{E}'=\mathcal{E}\cup\mathcal{E}_{\mathcal{P}}$, $\mathcal{D}'=\mathcal{D}(\mathcal{P})$.
   $\mathcal{S}$ and $\mathcal{S}'$ are equivalid by Theorem 4.11.
iii) $\mathcal{L}'=\mathcal{L}(\mathcal{R})$, $\mathcal{E}'=\mathcal{E}(\mathcal{R})$, $\mathcal{D}'=\mathcal{D}(\mathcal{R})$ where $\mathcal{R}$ is a canonical rewrite system equivalent to $\mathcal{E}$.
   By Theorem 4.12, $\mathcal{S}$ and $\mathcal{S}'$ are equivalid.

These cases prove that every two adjacent systems in the sequence of systems generated during the computation are equivalid and thus equisatisfiable. □

Theorem 4.15:
The EQUALITY LOOP RESIDUE PROCEDURE is complete.
Proof:
We have to show that any unsatisfiable system is recognized as unsatisfiable. Or, equivalently, if a system is recognized as satisfiable, it indeed is so.
It thus suffices to show that if the procedure terminates with the answer that the system is satisfiable that the obtained system is closed.

26: The stronger result is required in section 4.9 for further aspects and improvements.

Observe that the procedure LOOP_RESIDUE enumerates the admissible simple loops of the graph $\mathcal{G}(\mathcal{L})$ modulo cyclic permutation and reversal and that this is sufficient since by Corollaries A.2 and A.3 of appendix A, a loop $\mathcal{P}$ is an equality loop if and only if its reverse is an equality loop, and a permutable loop $\mathcal{P}$ is an equality loop if and only if any permutation of $\mathcal{P}$ is an equality loop.
Since the outermost REPEAT loop terminates for a satisfiable system if and only if $\mathcal{L}$ is a strictly closed set, condition iii) of the definition of a closed system is satisfied.
And because of the statement

$\mathcal{S}$:= REDUCTION($\mathcal{S}$, CANONIZATION($\mathcal{E}$))

conditions i) - ii) are also satisfied.
Therefore, the procedure terminates for a satisfiable system with a complete satisfiable system. In other words: the procedure is complete. □

4.7. Generalizations

Since a system is defined as a triple of lequalities, equalities, and inequalities, there are two ways of incorporating strict lequalities:
First, like in the second chapter, the notion of a strict loop with strict residue lequality can be appended, and second, $\mathcal{A}<\mathcal{B}$ can be replaced by $\mathcal{A}\leq\mathcal{B}$ and $\mathcal{A}\neq\mathcal{B}$.
The first option requires a modification of the procedure LOOP_RESIDUE. It has to distinguish between lequalities and strict lequalities and generates either a nonstrict loop residue lequality or a strict loop residue lequality, respectively.
The second option requires no modification in the algorithm: The basic version can already treat lequalities in this way. But instead, the disadvantage is that the information is not as compact as possible.
Nevertheless, we choose for convenience the second option for the examples of the following section.

The extension to an arbitrary number of terms per lequality is essentially in the same way as the extension to an arbitrary number of variables per lequality for the unextended class: By the use of symbolic computation.

4.8. Examples and observations

4.8.1. Examples

EXAMPLE 1:
In section 4.1, we considered the formula

$$\mathcal{F}_{fp} \equiv \quad [\, p(x) \supset x=2 \wedge f(2*z)=f(z) \wedge g(y)=x+7 \,] \supset [\, f(g(y))=f(5+2*x) \;\vee\; \neg p(x) \,].$$

By substituting the predicate symbol p by f_p, we obtain the formula

$$\mathcal{F}_f \equiv \quad [\, f_p(x)=0 \supset x=2 \wedge f(2*z)=f(z) \wedge g(y)=x+7 \,] \supset [\, f(g(y))=f(5+2*x) \;\vee\; \neg(f_p(x)=0) \,].$$

The disjunctive normal form of the negation has two conjuncts and results in the systems[27]

$$S_1 = (L_1, E_1, D_1) = (L, E, D \cup \{ f_p(x) \neq 0 \})$$
$$S_2 = (L_2, E_2, D_2) = (L, E \cup \{ x=2 \}, D)$$

where

$$L = \phi$$
$$E = \{ f(2*z)=f(z),\ g(y)-x=7,\ f_p(x)=0 \}$$
$$D = \{ f(g(y)) \neq f(5+2*x) \}.$$

Since L_1 and L_2 are empty, the procedure LOOP_RESIDUE does not enumerate any simple admissible loop in $G(L_1)$ or $G(L_2)$ and therefore, the innermost and outermost REPEAT loops terminate immediately.
For the system S_1,

$$\text{CANONIZATION}(E_1) = \{ f(2*z) \rightarrow f(z),\ g(y) \rightarrow x+7,\ f_p(x) \rightarrow 0 \}.$$

This reduces $D_1 = \{ f(g(y)) \neq f(5+2*x),\ f_p(x) \neq 0 \}$ to

$$D_1(\text{CANONIZATION}(E_1)) = \{ f(x+7) \neq f(5+2*x),\ 0 \neq 0 \}.$$

27: For convenience, the formulas $at+bt'\ rel\ c$ where $rel \in \{\leq, =, <\}$ are given in a non standardized manner. For example, the formula $0*t_0+1*f(x) \neq 0$ is written as $f(x) \neq 0$.

Since this resulting set of inequalities contains $0 \neq 0$, the system $\mathcal{S}_1$ is unsatisfiable.

In the same way, for the system $\mathcal{S}_2$,

$$\begin{aligned} &\text{CANONIZATION } (\mathcal{E}_2) \\ = \ &\text{CANONIZATION } (\{ f(2*2)=f(2), \\ &\qquad\qquad g(y)-x=7, \\ &\qquad\qquad f_{\mathcal{P}}(x)=0 \\ &\qquad\qquad x=2 \}) \\ = \ &\{ f(2*2) \to f(2), \\ &\quad g(y) \to 9, \\ &\quad f_{\mathcal{P}}(2) \to 0, \\ &\quad x \to 2 \} \end{aligned}$$

which reduces $\mathcal{D}_2 = \{f(g(y)) \neq f(5+2*x)\}$ to $\{0 \neq 0\}$ with intermediate steps

$\{ f(9) \neq f(5+2*x) \}$,

and $\{ f(9) \neq f(9) \}$.

Hence, the system $\mathcal{S}_2$ is also unsatisfiable.

EXAMPLE 2:

Reconsider the formula of section 4.2.3,

$$\mathcal{F} \equiv (x \leq gx \wedge gx \leq x \supset x = ggggggx).$$

The negation of this formula results in one system $\mathcal{S}$ where

$$\mathcal{S} = (\{ x \leq gx, gx \leq x \}, \phi, \{ x \neq ggggggx \}).$$

The procedure LOOP_RESIDUE enumerates one loop, the equality loop $\mathcal{P}$, which is labeled with the lequality set $\mathcal{L}_{\mathcal{P}} = \mathcal{L}_{\mathcal{P}_{\leq}} = \{x \leq gx, gx \leq x\}$ and which has residue $\langle 1, -1, 0 \rangle$ independently of the initial vertex. By the definition of $\mathcal{E}_{\mathcal{P}}$ and $\mathcal{S}(\mathcal{P})$, we have

$$\mathcal{E}_{\mathcal{P}} = \{ 1*x+(-1)*gx = 0 \} = \{x=gx\},$$

$$\mathcal{R}(\mathcal{E}_{\mathcal{P}}) = \{gx \to x\},$$

and $$\mathcal{S}(\mathcal{P}) = (\phi, \{x=gx\}, \{0 \neq 0\}).$$

This proves the unsatisfiabiliy of $\mathcal{S}$.

EXAMPLE 3:

Let us now reconsider the instructive formula of section 4.3,

$$\mathcal{F} \equiv \{ (f(a,c) \le a \wedge a \le f(a,c)) \vee (x \le g(x) \wedge g(x) \le x) \vee (b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b) \} \supset \{ (f(f(a,c),c) \le a \wedge a \le f(f(a,c),c)) \vee (g^5(x) \le x \wedge x \le g^5(x)) \vee (h(b) \le b \wedge b \le h(b)) \}.$$

The negated formula

$$\neg\mathcal{F} \equiv \{ (f(a,c) \le a \wedge a \le f(a,c)) \vee (x \le g(x) \wedge g(x) \le x) \vee (b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b) \} \wedge \neg \{ (f(f(a,c),c) \le a \wedge a \le f(f(a,c),c)) \vee (g^5(x) \le x \wedge x \le g^5(x)) \vee (h(b) \le b \wedge b \le h(b)) \}$$

$$\equiv \{ (f(a,c) \le a \wedge a \le f(a,c)) \vee (x \le g(x) \wedge g(x) \le x) \vee (b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b) \} \wedge \{ (f(f(a,c),c) > a \vee a > f(f(a,c),c)) \wedge (g^5(x) > x \vee x > g^5(x)) \wedge (h(b) > b \vee b > h(b)) \}.$$

results in 24 (3*2*2*2) systems each of which contains lequalities that form a simple equality loop $\mathcal{P}$ where $\mathcal{E}_{\mathcal{P}}$ contains either $a=f(a,c)$, $x=g(x)$, or $b=h^3(b)$ and $b=h^5(b)$. Each system requires one call to LOOP_RESIDUE. The equalities are obtained by one call of the procedure LOOP_COLLABATION. For the first two equalities, the inequality $0\neq 0$ is obtained by the same call. And for $b=h^3(b)$ and $b=h^5(b)$, one call of each of the procedures CANONIZATION and REDUCTION is required to obtain $0\neq 0$.

Observe that this example combines the three valid formulas

$$f(a,c) \le a \wedge a \le f(a,c) \supset f(f(a,c),c) \le a \wedge a \le f(f(a,c),c),$$

$$x \le g(x) \wedge g(x) \le x) \supset g^5(x) \le x \wedge x \le g^5(x),$$

and $$b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b) \supset h(b) \le b \wedge b \le h(b)$$

two of which are similiar to the examples of section 3.3,

$$f(a,b)=a \supset f(f(a,b),b)=a,$$

and $$fffff a=a \wedge fff a=a \supset f a=a.$$

EXAMPLE 4:

Consider the system $\mathcal{S} = (\mathcal{L}, \phi, \phi)$ for

$$
\begin{aligned}
\mathcal{L} = \{ \quad (1) \quad & a \le 2b - 1, \\
(2) \quad & 4b \le f(a + 2d + 5e) + 3, \\
(3) \quad & f(a + 2d + 5e) \le 2a - 1, \\
(4) \quad & c \le 2d + 1, \\
(5) \quad & 10d \le f(2b + c - g(x)) - 10, \\
(6) \quad & f(2b + c - g(x)) \le 5c + 5, \\
(7) \quad & 5e \le -g(x) + 2, \\
(8) \quad & -g(x) \le f(y) - 7, \\
(9) \quad & f(y) \le 5e + 5, \\
(10) \quad & f(y) \le f(g(u)) \\
(11) \quad & f(g(u)) \le 2f(a + 2d + 5e) + 5, \\
(12) \quad & f(2b + c - g(x)) \le 2f(g(v)) - 4, \\
(13) \quad & 4f(g(v)) \le -g(x) + 9 \qquad \}.
\end{aligned}
$$

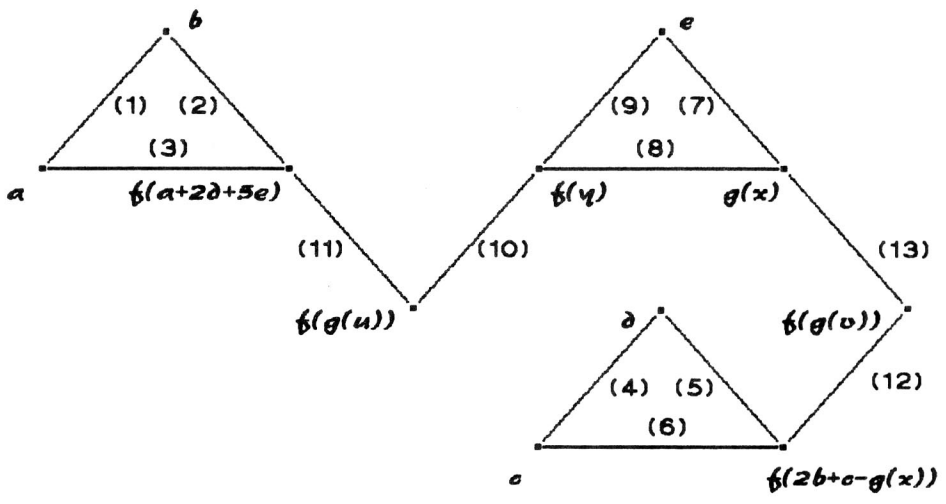


Figure 4.6.1: Graph $\mathcal{G}(\mathcal{L})$

There are three admissible simple loops $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ in the graph $\mathcal{G}(\mathcal{L})$ labeled by the lequalities (1) - (3), (4) - (6), and (7) - (9). Their residues are

$$r(\mathcal{P}_1) = (\langle 1, -2, -1\rangle * \langle 4, -1, 3\rangle) * \langle 1, -2, -1\rangle$$
$$= \langle 4, -2, 2\rangle * \langle 1, -2, -1\rangle$$
$$= \langle 4, -4, 0\rangle,$$
$$r(\mathcal{P}_2) = (\langle 1, -2, 1\rangle * \langle 10, -1, -10\rangle) * \langle 1, -5, 5\rangle$$
$$= \langle 10, -2, -10\rangle * \langle 1, -5, 5\rangle$$
$$= \langle 10, -10, 0\rangle,$$

and

$$r(\mathcal{P}_3) = (\langle 5, 1, 2\rangle * \langle -1, -1, -7\rangle) * \langle 1, -5, 5\rangle$$
$$= \langle 5, -1, -5\rangle * \langle 1, -5, 5\rangle$$
$$= \langle 5, -5, 0\rangle.$$

We assume that these equality loops are encountered in the ordering $\mathcal{P}_3$, $\mathcal{P}_1$, and $\mathcal{P}_2$ and that the following chain holds for the total simplification ordering [28]:

$$a \prec b \prec c \prec d \prec e \prec u \prec v \prec x \prec y \prec f \prec g$$

We then obtain a sequence $\langle \mathcal{S}=\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\rangle$ of systems. For the loop $\mathcal{P}_3$, and the resulting system $\mathcal{S}_1$, we have

$$\mathcal{E}(\mathcal{P}_3) = \{\ 5e + g(x) = 2,\quad 5e - f(y) = -5\ \}$$
$$\mathcal{R}(\mathcal{E}(\mathcal{P}_3)) = \{\ g(x) \rightarrow -5e + 2,\quad f(y) \rightarrow 5e + 5\ \}$$

$$\mathcal{L}_1 = \{$$
$$(1)\quad a \le 2b - 1,$$
$$(2)\quad 4b \le f(a + 2d + 5e) + 3,$$
$$(3)\quad f(a + 2d + 5e) \le 2a - 1,$$
$$(4)\quad c \le 2d + 1,$$
$$(5)\quad 10d \le f(2b + c + 5e - 2) - 10,$$
$$(6)\quad f(2b + c + 5e - 2) \le 5c + 5,$$
$$(10)\quad 5e \le f(g(u)) - 5,$$
$$(11)\quad f(g(u)) \le 2f(a + 2d + 5e) + 5,$$
$$(12)\quad f(2b + c + 5e - 2) \le 2f(g(v)) - 4,$$
$$(13)\quad 4f(g(v)) \le 5e + 7\qquad \}.$$

The graph $\mathcal{G}(\mathcal{L}_1)$ is shown in figure 4.6.2.
Observe that the loop collabation of the equality loop $\mathcal{P}_3$ in the graph $\mathcal{G}(\mathcal{L}_0)=\mathcal{G}(\mathcal{L})$ did not affect the residues and subresidues of the loops $\mathcal{P}_1$ and $\mathcal{P}_2$ in the graph $\mathcal{G}(\mathcal{L}_1)$.
Since

28: Recall that the variables occurring in the system can be regarded as constants since the formula can be skolemized to an equisatisfiable ground formula.

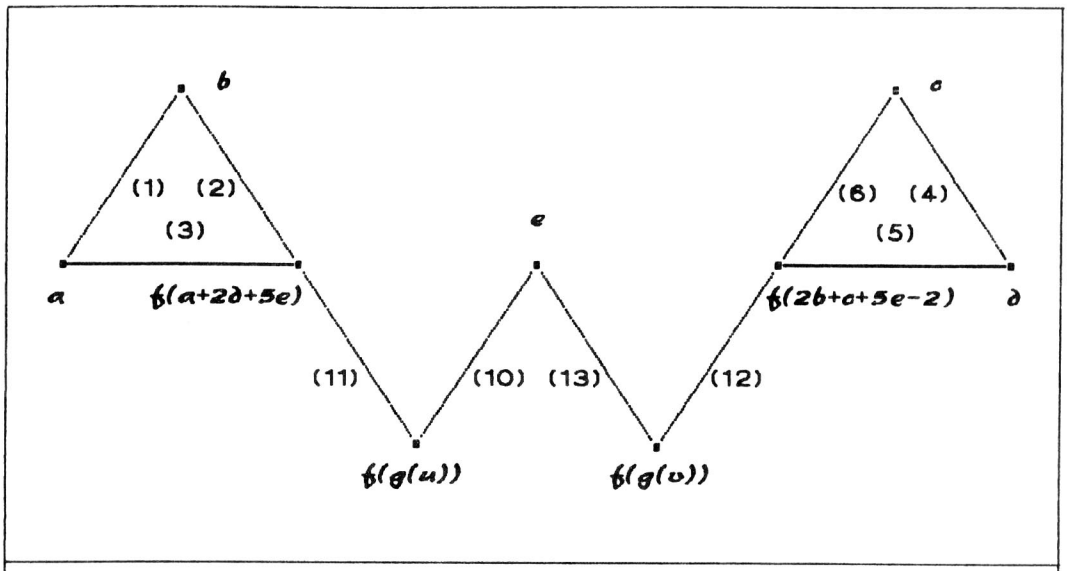


Figure 4.6.2: Graph $G(\mathcal{L}_1)$

$$\mathcal{E}(\mathcal{P}_1) = \{ \quad a - 2b = -1, \quad 4a - 2f(a + 2\partial + 5e) = 2 \quad \}$$

and

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_1)) = \{ \; b \to (1/2)a + (1/2), \quad f(a + 2\partial + 5e) \to 2a - 1 \; \},$$

the call of procedure LOOP_COLLABATION for arguments $\mathcal{S}_1$ and $\mathcal{P}_1$ results in the system $\mathcal{S}_2$ where

$$\begin{aligned}
\mathcal{L}_2 = \{ \quad (4) \quad & c \le 2\partial + 1, \\
(5) \quad & 10\partial \le f(a + c + 5e - 1) - 10, \\
(6) \quad & f(a + c + 5e - 1) \le 5c + 5, \\
(10) \quad & 5e \le f(g(u)) - 5, \\
(11) \quad & f(g(u)) \le 4a + 3, \\
(12) \quad & f(a + c + 5e - 1) \le 2f(g(v)) - 4, \\
(13) \quad & 4f(g(v)) \le 5e + 7 \quad \}.
\end{aligned}$$

The graph for the set $\mathcal{L}_2$ of lequalities is shown in figure 4.6.3.

Finally, we have for the loop $\mathcal{P}_2$

$$\mathcal{E}(\mathcal{P}_2) = \{ \quad c - 2\partial = 1, \quad 10c - 2f(a + c + 5e - 1) = -10 \quad \}$$

and

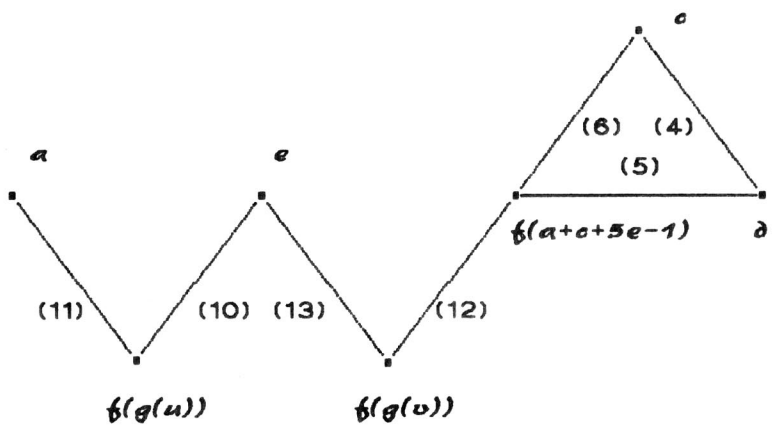


Figure 4.6.3: Graph $G(L_2)$

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_2)) = \{ \quad \partial \rightarrow (1/2)c - (1/2), \quad f(a + c + 5e - 1) \rightarrow 5c + 5 \quad \}.$$

The procedure LOOP_COLLABATION computes S_3 where

$$L_3 = \{ \begin{array}{ll} (10) & 5e \leq f(g(u)) - 5, \\ (11) & f(g(u)) \leq 4a + 3, \\ (12) & 5c \leq 2f(g(v)) - 9, \\ (13) & 4f(g(v)) \leq 5e + 7 \end{array} \}$$

and

$$\mathcal{E}_3 = \{ \begin{array}{ll} 5e + g(x) & = 2, \\ 5e - f(y) & = -5, \\ a - 2b & = -1, \\ 4a - 2f(a + 2\partial + 5e) & = 2, \\ c - 2\partial & = 1, \\ 10c - 2f(a + c + 5e - 1) & = -10 \end{array} \}.$$

Now the innermost REPEAT loop terminates and the flag *collapsed* is set. We now obtain the system S_4 by the procedures CANONIZATION, which computes a canonical rewriting system $\mathcal{R}(\mathcal{E}_4)$ equivalent to the set $\mathcal{E}_3$ of equalities, and REDUCTION, which reduces the lequalities in L_3 to obtain L_4. The resulting system $S_4=(L_4, \mathcal{E}_4, \mathcal{D}_4)$ has the components

$$\mathcal{E}_4 = \{ \begin{array}{ll} 5e + g(x) & = 2, \\ 5e - f(y) & = -5, \\ a - 2b & = -1, \\ 4a - 10c & = 12, \\ 4a - 20d & = 22, \\ 4a - 2f((7/5)a + 5e - (11/5)) & = 2 \quad \}. \end{array}$$

and

$$\mathcal{L}_4 = \{ \begin{array}{ll} (10) & 5e \le f(g(u)) - 5, \\ (11) & f(g(u)) \le 4a + 3, \\ (12) & 2a \le 2f(g(v)) - 3, \\ (13) & 4f(g(v)) \le 5e + 7 \quad \}. \end{array}$$

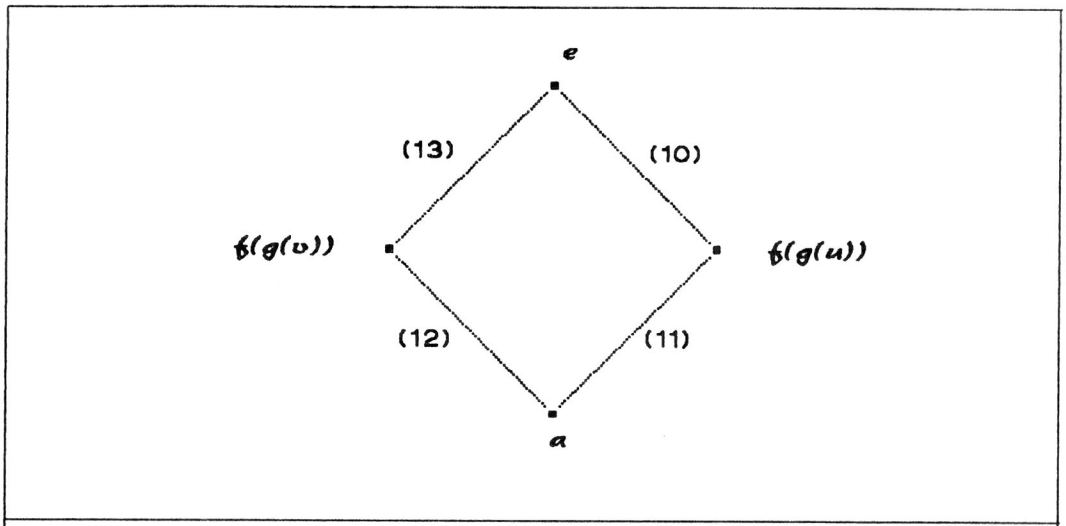


Figure 4.6.4: Graph $\mathcal{G}(\mathcal{L}_3)$

The lequalities of the set $\mathcal{L}_4$ label an admissible simple loop $\mathcal{P}_4$ in the graph $\mathcal{G}(\mathcal{L}_4)$ which has the residue

$$\begin{aligned} r(\mathcal{P}_4) &= (\langle 5, -1, -5\rangle * \langle 1, -4, 3\rangle) * (\langle 2, -2, -3\rangle * \langle 4, -5, 7\rangle) \\ &= \langle 5, -4, -2\rangle * \langle 8, -10, 2\rangle \\ &= \langle 40, -40, -8\rangle . \end{aligned}$$

Since $\mathcal{P}_4$ is thus an infeasible simple loop, the system $\mathcal{S}$ also is infeasible. Note that for this example, the computation process would have been similiar for any other ordering of the loops since the vertices of the graph $\mathcal{G}(\mathcal{L})$ labeled by $f(a + 2d + 5e)$ and $f(2b + c - g(x))$ can only coincide after all three equality loops are collapsed.

EXAMPLE 5:

To demonstrate the effect of the procedure LOOP_COLLABATION wrt the variation of the structure of the graph for the lequalities, let us now consider the system $\mathcal{S}=(\mathcal{L},\mathcal{E},\mathcal{D})$ where

$$
\begin{aligned}
\mathcal{L} = \{\ (1)\quad & 3t \le u + 2,\\
(2)\quad & u \le 3w - 2,\\
(3)\quad & 3w \le v + 2,\\
(4)\quad & v \le 3t - 2,\\
(5)\quad & 2f(3g(t) + 2h(u)) \le h(a) + 1,\\
(6)\quad & 2h(a) \le 5t - 2,\\
(7)\quad & u \le 3g(b) - 8,\\
(8)\quad & 5g(b) \le 4f(3g(w) + 2h(v)) + 10,\\
(9)\quad & 5g(-4g(b) + 2h(a) + 12) \le 6h(a) + 6,\\
(10)\quad & v \le f(c) - 3,\\
(11)\quad & f(c) \le g(6g(b) - 2h(a) - 12) + 1,\\
(12)\quad & 2h(2u - f(c) + 9) \le 3f(\partial) + 5,\\
(13)\quad & 3f(\partial) \le w - 5,\\
(14)\quad & g(b) \le 2h(-u + 2f(c)) + 2,\\
(15)\quad & f(c) \le 3f(3f(\partial)) + 1,\\
(16)\quad & f(t - 5) \le 3f(\partial) + 4 \qquad \}
\end{aligned}
$$

and $\mathcal{E}=\mathcal{D}=\phi$. The graph $\mathcal{G}(\mathcal{L})=\mathcal{G}(\mathcal{L}_0)$ is shown in figure 4.7.1.

We demonstrate now how procedure 4.5 computes a sequence $<\mathcal{S}=\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_4>$ of systems. Let $\prec$ be a total simplification ordering with

$$a \prec b \prec c \prec \partial \prec t \prec u \prec v \prec w \prec f \prec g \prec h.$$

We will see that the procedure LOOP_COLLABATION is so effective such that all computation is done within the innermost REPEAT-loop which thus is only encountered once.

There is only one admissible simple loop $\mathcal{P}_0$ in the graph $\mathcal{G}(\mathcal{L}_0)=\mathcal{G}(\mathcal{L})$. It is labeled by the lequalities (1), (2), (3), and (4). Its residue computes to

$$
\begin{aligned}
r(\mathcal{P}_0) &= ((<3,-1,2> * <1,-3,-2>) * <3,-1,2>) * <1,-3,-2>\\
&= (<3,-3,0> * <3,-1,2>) * <1,-3,-2>\\
&= <9,-3,6> * <1,-3,-2>\\
&= <9,-9,0>.
\end{aligned}
$$

The loop $\mathcal{P}_0$ is therefore an equality simple loop. We have for this loop

$\langle a_{11}, b_{11}, c_{11} \rangle = \langle 3, -1, 2 \rangle$

$\langle a_{12}, b_{12}, c_{12} \rangle = \langle 3, -3, 0 \rangle$

$\langle a_{13}, b_{13}, c_{13} \rangle = \langle 9, -3, 6 \rangle$

and therefore

$$\mathcal{E}(\mathcal{P}_0) = \{\ 3t + (-1)u = 2,\ 3t + (-3)w = 0,\ 9t + (-3)v = 6\ \}$$

and

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_0)) = \{\ u \to 3t - 2,\ w \to t,\ v \to 3t - 2\ \}.$$

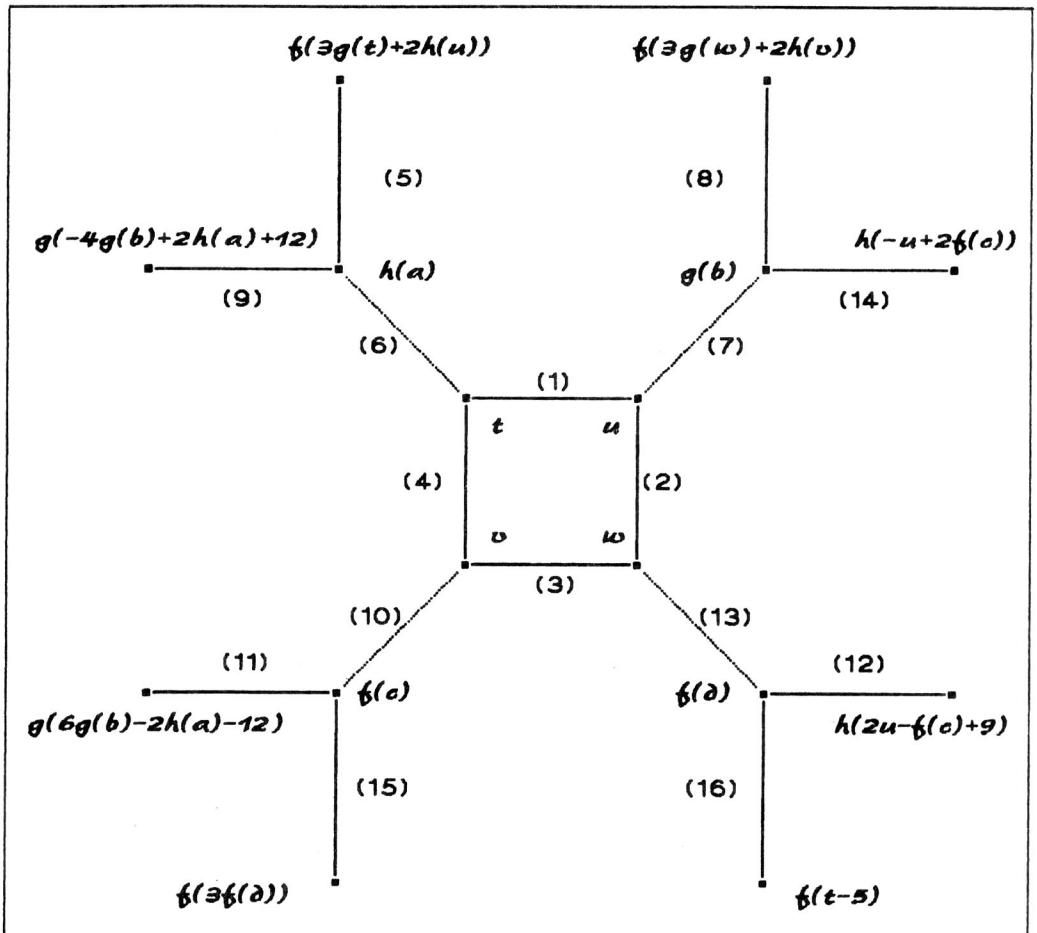


Figure 4.7.1: Graph $\mathcal{G}(\mathcal{L}_0)$

The procedure LOOP_COLLABATION reduces the lequalities (5), (7), (8), (10), (12), (13) and (14), and computes the system $S_1=(L_1,E_1,D_1)$ where

$$
\begin{aligned}
L_1 = \{ (5)\quad & 2f(3g(t) + 2h(3t - 2)) \le h(a) + 1,\\
(6)\quad & 2h(a) \le 5t - 2,\\
(7)\quad & 3t \le 3g(b) - 6,\\
(8)\quad & 5g(b) \le 4f(3g(t) + 2h(3t - 2)) + 10,\\
(9)\quad & 5g(-4g(b) + 2h(a) + 12) \le 6h(a) + 6,\\
(10)\quad & 3t \le f(c) - 1,\\
(11)\quad & f(c) \le g(6g(b) - 2h(a) - 12) + 1,\\
(12)\quad & 2h(6t - f(c) + 5) \le 3f(\partial) + 5,\\
(13)\quad & 3f(\partial) \le t - 5,\\
(14)\quad & g(b) \le 2h(-3t + 2f(c) + 2) + 2,\\
(15)\quad & f(c) \le 3f(3f(\partial)) + 1,\\
(16)\quad & f(t - 5) \le 3f(\partial) + 4 \qquad \},
\end{aligned}
$$

$$E_1 = E(P_0)$$

and $D_1=\phi$. The graph G for L_1 is shown in figure 4.7.2.

The reduction wrt the rewrite rules of the loop P_0 in the graph $G(L_0)$ has produced a new admissible simple loop P_1 in the graph $G(L_1)$ labeled with the lequalities (7), (8), (5), and (6) and with residue

$$
\begin{aligned}
r(P_1) &= ((\langle 3, -3, -6\rangle * \langle 5, -4, 10\rangle) * \langle 2, -1, 1\rangle) * \langle 2, -5, -2\rangle\\
&= (\langle 15, -12, 0\rangle * \langle 2, -1, 1\rangle) * \langle 2, -5, -2\rangle\\
&= \langle 30, -12, 12\rangle * \langle 2, -5, -2\rangle\\
&= \langle 60, -60, 0\rangle .
\end{aligned}
$$

The simple equality loop P_1 has subresidues

$$\langle a_{11}, b_{11}, c_{11}\rangle = \langle 3, -3, -6\rangle$$

$$\langle a_{12}, b_{12}, c_{12}\rangle = \langle 15, -12, 0\rangle$$

$$\langle a_{13}, b_{13}, c_{13}\rangle = \langle 30, -12, 12\rangle$$

and therefore

$$
\begin{aligned}
E(P_1) = \{ \quad & 3t + (-3)g(b) = -6,\\
& 15t + (-12)f(3g(t) + 2h(3t - 2)) = 0,\\
& 30t + (-12)h(a) = 12 \}
\end{aligned}
$$

with

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_1)) = \{\ g(b) \to t + 2,\quad f(3g(t) + 2h(3t - 2)) \to (5/4)t,\quad h(a) \to (5/2)t - 1\ \}.$$

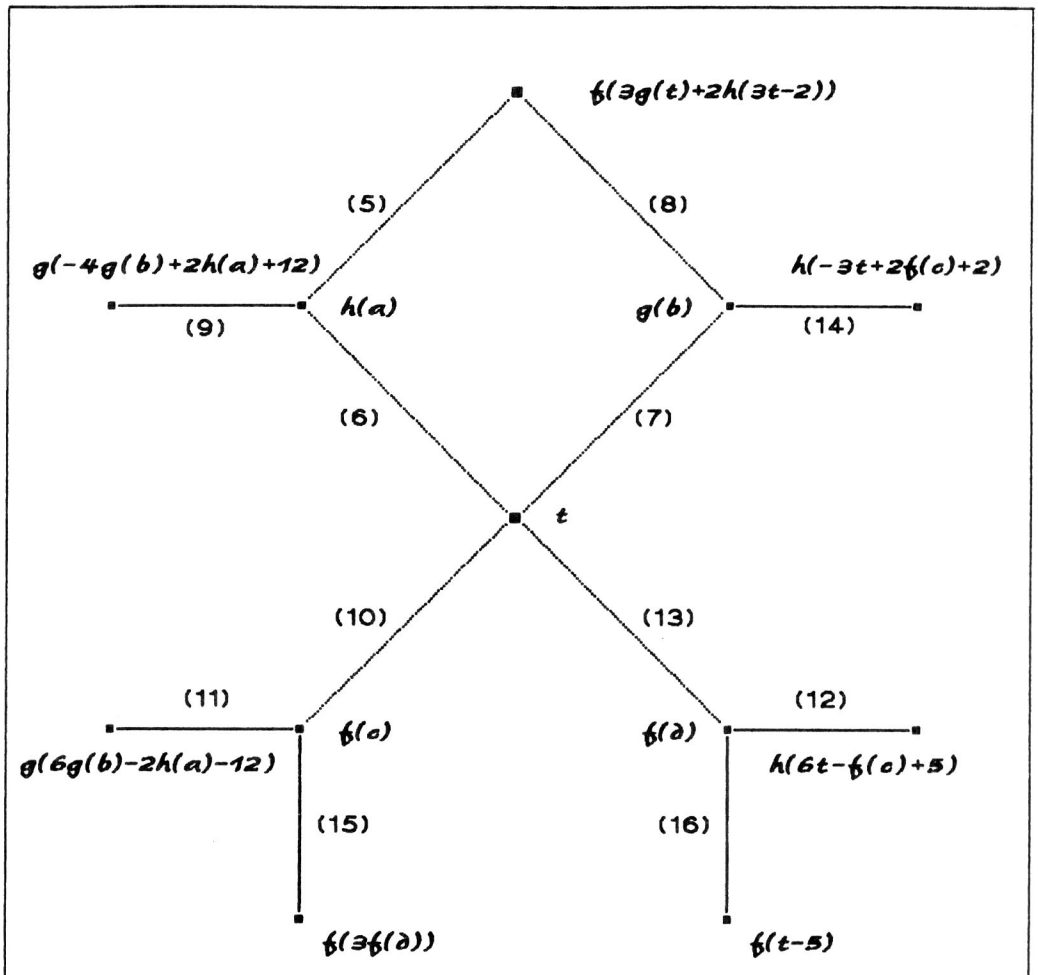


Figure 4.7.2: Graph $\mathcal{G}(\mathcal{L}_1)$

The procedure LOOP_COLLABATION is called for the arguments $\mathcal{S}_1$ and $\mathcal{P}_1$, and both

$$g(-4g(b) + 2h(a) + 12) \quad (\to^* \quad g(-4(t + 2) + 2((5/2)t - 1) + 12)\ )$$

and

$$g(6g(b) - 2h(a) - 12) \quad (\to^* \quad g(6(t + 2) - 2((5/2)t - 1) - 12)\ )$$

are reduced to

$$g(t+2).$$

in the lequalities (9) and (11).
Similiarly, $\mathcal{R}(\mathcal{E}(\mathcal{P}_1))$ reduces $6h(a)+6$ to $15t$ in lequality (9) and we obtain the system $\mathcal{S}_2=(\mathcal{L}_2,\mathcal{E}_2,\mathcal{D}_2)$ where

$$\mathcal{L}_2 = \{$$
$$(9)\quad 5g(t+2) \le 15t,$$
$$(10)\quad 3t \le f(c)-1,$$
$$(11)\quad f(c) \le g(t+2)+1,$$
$$(12)\quad 2h(6t-f(c)+5) \le 3f(\partial)+5,$$
$$(13)\quad 3f(\partial) \le t-5,$$
$$(14)\quad t \le 2h(-3t+2f(c)+2),$$
$$(15)\quad f(c) \le 3f(3f(\partial))+1,$$
$$(16)\quad f(t-5) \le 3f(\partial)+4 \quad \},$$

$$\mathcal{E}_2 = \mathcal{E}_1 \cup \mathcal{E}(\mathcal{P}_1),$$

and $\mathcal{D}_2=\phi$. The graph $\mathcal{G}$ for $\mathcal{L}_2$ is shown in figure 4.7.3.
Again, an admissible simple loop $\mathcal{P}_2$ is encountered in the graph $\mathcal{G}(\mathcal{L}_2)$. It is labeled by the lequalities (10), (11), and (9), and its residue is

$$\begin{aligned} r(\mathcal{P}_2) &= (\langle 3,-1,-1\rangle * \langle 1,-1,1\rangle) * \langle 5,-15,0\rangle \\ &= \langle 3,-1,0\rangle * \langle 5,-15,0\rangle \\ &= \langle 15,-15,0\rangle . \end{aligned}$$

The loop $\mathcal{P}_2$ is thus an equality loop and has subresidues

$$\langle a_{11}, b_{11}, c_{11}\rangle = \langle 3,-1,-1\rangle$$
$$\langle a_{12}, b_{12}, c_{12}\rangle = \langle 3,-1,0\rangle$$

and therefore

$$\mathcal{E}(\mathcal{P}_2) = \{\ 3t+(-1)f(c) = -1,\quad 3t+(-1)g(t+2) = 0\ \}$$

with

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_2)) = \{\ f(c) \to 3t+1,\quad g(t+2) \to 3t\ \}.$$

The terms $h(6t-f(c)+5)$ and $h(-3t+2f(c)+2)$ are both rewritten to $h(3t+4)$ and procedure LOOP_COLLABATION reduces $\mathcal{S}_2$ wrt $\mathcal{R}(\mathcal{E}(\mathcal{P}_2))$ to the system $\mathcal{S}_3=(\mathcal{L}_3,\mathcal{E}_3,\mathcal{D}_3)$ where

$$\mathcal{L}_3 = \{ \quad (12) \quad 2h(3t+4) \le 3f(\partial) + 5,$$
$$(13) \quad 3f(\partial) \le t - 5,$$
$$(14) \quad t \le 2h(3t+4),$$
$$(15) \quad t \le f(3f(\partial)),$$
$$(16) \quad f(t-5) \le 3f(\partial) + 4 \quad \},$$

$\mathcal{E}_3 = \mathcal{E}_2 \cup \mathcal{E}(\mathcal{P}_2)$,

and $\mathcal{D}_3 = \phi$. The graph $\mathcal{G}$ for $\mathcal{L}_3$ is shown in figure 4.7.4 and contains

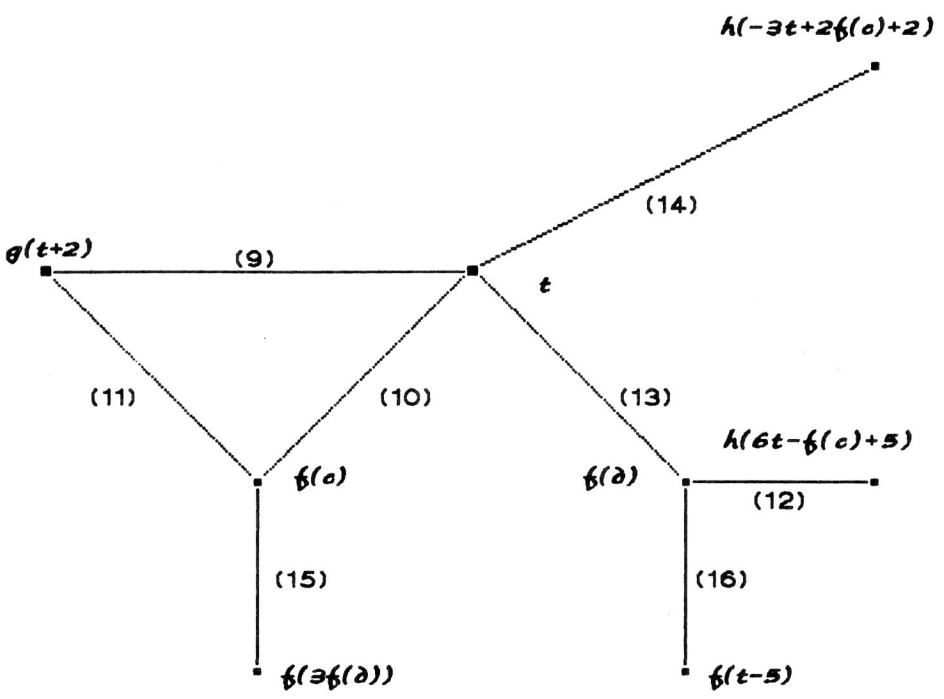


Figure 4.7.3: Graph $\mathcal{G}(\mathcal{L}_2)$

an admissible simple loop $\mathcal{P}_3$ labeled by the lequalities (14), (12), and (13) and residue

$$r(\mathcal{P}_3) = (\langle 1, -2, 0\rangle * \langle 2, -3, 5\rangle) * \langle 3, -1, -5\rangle$$
$$= \langle 2, -6, 10\rangle * \langle 3, -1, -5\rangle$$
$$= \langle 6, -6, 0\rangle .$$

The equality loop $\mathcal{P}_3$ has thus subresidues

$\langle a_{11}, b_{11}, c_{11} \rangle = \langle 1, -2, 0 \rangle$

$\langle a_{12}, b_{12}, c_{12} \rangle = \langle 2, -6, 10 \rangle$

and we obtain

$$\mathcal{E}(\mathcal{P}_3) = \{ \; 1t + (-2)h(3t + 4) = 0, \quad 2t + (-6)f(\partial) = 10 \; \}$$

and

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_3)) = \{ \; h(3t + 4) \rightarrow (1/2)t, \quad f(\partial) \rightarrow (1/3)t - (5/3) \; \}.$$

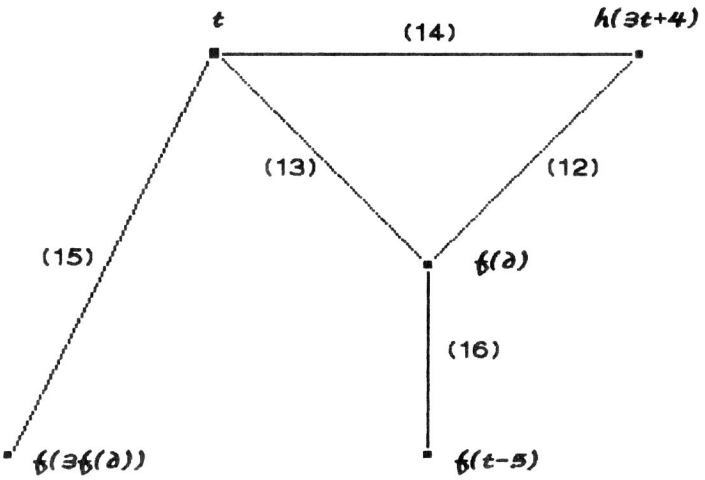


Figure 4.7.4: Graph $\mathcal{G}(\mathcal{L}_3)$

We finally obtain the system $\mathcal{S}_4 = (\mathcal{L}_4, \mathcal{E}_4, \mathcal{D}_4)$ from the procedure LOOP_COLLABATION which is called for arguments $\mathcal{S}_3$ and $\mathcal{E}(\mathcal{P}_3)$. We have

$$\mathcal{L}_4 = \{ \; (15) \quad t \leq f(t - 5), \quad (16) \quad f(t - 5) \leq t - 1 \; \},$$

and an admissible simple loop $\mathcal{P}_4$ labeled by the two lequalities (15) and (16) in the graph $\mathcal{G}(\mathcal{L}_4)$. It is infeasible since

$$r(\mathcal{P}_4) = \langle 1, -1, 0 \rangle * \langle 1, -1, -1 \rangle = \langle 1, -1, -1 \rangle.$$

The unsatisfiability of the given system is now returned from the innermost REPEAT loop.

EXAMPLE 6:

Let us now look at an example for which the ordering of the encountered equality loops directs the construction of the sequence of systems.

Consider the system $S=(\mathcal{L}, \phi, \phi)$ where

$$
\begin{aligned}
\mathcal{L} = \{ \quad (1) \quad & 4x - 7\lfloor(2a + 1)\rfloor \le 0, \\
(2) \quad & 7\lfloor(3b + 2)\rfloor - 4x \le -1, \\
(3) \quad & 7\lfloor(3b + 2)\rfloor - 7\lfloor(2a + 1)\rfloor \le -1, \\
(4) \quad & 2a - 7\lfloor(2a + 1)\rfloor \le -1, \\
(5) \quad & 7\lfloor(2a + 1)\rfloor - 5y \le 0, \\
(6) \quad & 5y - 7\lfloor(3b + 2)\rfloor \le 1, \\
(7) \quad & 7\lfloor(3b + 2)\rfloor - 3b \le 1, \\
(8) \quad & 5y - 2a \le 1, \\
(9) \quad & 3b - 5y \le -2 \quad \}.
\end{aligned}
$$

The graph $G(\mathcal{L})$, shown in figure 4.8.1, has four admissible simple loops

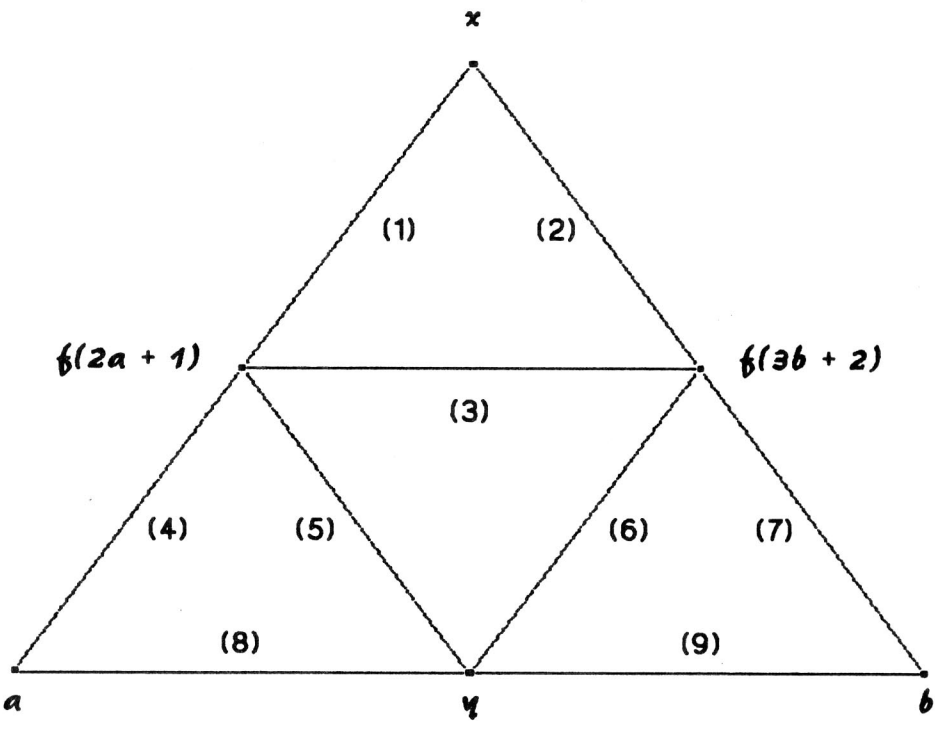


Figure 4.8.1.: Graph $G(\mathcal{L})$

$\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ which are labeled by the lequalities

$$\begin{aligned}
\mathcal{L}(\mathcal{P}_1) = \{ \quad & (4) \quad 2a - 7\lfloor(2a+1)\rfloor \le -1, \\
& (5) \quad 7\lfloor(2a+1)\rfloor - 5y \le 0, \\
& (8) \quad 5y - 2a \le 1 \quad \}, \\
\mathcal{L}(\mathcal{P}_2) = \{ \quad & (6) \quad 5y - 7\lfloor(3b+2)\rfloor \le 1, \\
& (3) \quad 7\lfloor(3b+2)\rfloor - 7\lfloor(2a+1)\rfloor \le -1, \\
& (5) \quad 7\lfloor(2a+1)\rfloor - 5y \le 0 \quad \}, \\
\mathcal{L}(\mathcal{P}_3) = \{ \quad & (9) \quad 3b - 5y \le -2, \\
& (6) \quad 5y - 7\lfloor(3b+2)\rfloor \le 1, \\
& (7) \quad 7\lfloor(3b+2)\rfloor - 3b \le 1 \quad \},
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{L}(\mathcal{P}_4) = \{ \quad & (1) \quad 4x - 7\lfloor(2a+1)\rfloor \le 0, \\
& (5) \quad 7\lfloor(2a+1)\rfloor - 5y \le 0, \\
& (6) \quad 5y - 7\lfloor(3b+2)\rfloor \le 1, \\
& (2) \quad 7\lfloor(3b+2)\rfloor - 4x \le -1 \quad \}.
\end{aligned}$$

Their residues are

$$\begin{aligned}
r(\mathcal{P}_1) &= \langle 2, -7, -1\rangle * \langle 7, -5, 0\rangle * \langle 5, -2, 1\rangle \\
&= \langle 14, -35, -7\rangle * \langle 5, -2, 1\rangle \\
&= \langle 70, -70, 0\rangle, \\
r(\mathcal{P}_2) &= \langle 5, -7, 1\rangle * \langle 7, -7, -1\rangle * \langle 7, -5, 0\rangle \\
&= \langle 35, -49, 0\rangle * \langle 7, -5, 0\rangle \\
&= \langle 245, -245, 0\rangle, \\
r(\mathcal{P}_3) &= \langle 3, -5, -2\rangle * \langle 5, -7, 1\rangle * \langle 7, -3, 1\rangle \\
&= \langle 15, -35, -5\rangle * \langle 7, -3, 1\rangle \\
&= \langle 105, -105, 0\rangle,
\end{aligned}$$

and

$$\begin{aligned}
r(\mathcal{P}_4) &= \langle 4, -7, 0\rangle * \langle 7, -5, 0\rangle * \langle 5, -7, 1\rangle * \langle 7, -4, -1\rangle \\
&= \langle 28, -35, 0\rangle * \langle 5, -7, 1\rangle * \langle 7, -4, -1\rangle \\
&= \langle 140, -245, 35\rangle * \langle 7, -4, -1\rangle \\
&= \langle 980, -980, 0\rangle.
\end{aligned}$$

So we have four simple equality loops.

We now construct three sequences of systems by chosing the short loop $\mathcal{P}_1$ containing a vertex with lower outdegree (the vertex labeled by a incidents

only with two edges) for the first sequence, the central loop $\mathcal{P}_2$ for the second sequence, and the large loop $\mathcal{P}_4$ for the third sequence.
The ordering $\prec$ may w. l. o. g. satisfy $a \prec b \prec x \prec y \prec f$.
The subresidues, the equalities, and the corresponding rewrite rules obtained by the equality loop $\mathcal{P}_1$ are

$$\langle a_{11}, b_{11}, c_{11} \rangle = \langle 2, -7, -1 \rangle$$
$$\langle a_{12}, b_{12}, c_{12} \rangle = \langle 14, -35, -7 \rangle,$$
$$\mathcal{E}(\mathcal{P}_1) = \{ \quad 2a - 7f(2a + 1) = -1, \quad 14a - 35y = -7 \quad \},$$
$$\mathcal{R}(\mathcal{E}(\mathcal{P}_1)) = \{ \quad f(2a + 1) \rightarrow (2/7)a + (1/7), \quad y \rightarrow (2/5)a + (1/5) \quad \}.$$

By calling the procedure LOOP_COLLABATION for $\mathcal{S}$ and $\mathcal{P}_1$, we obtain the system $\mathcal{S}_1 = (\mathcal{L}_1, \mathcal{E}(\mathcal{P}_1), \emptyset)$ where the set $\mathcal{L}_1$ is given by

$$\mathcal{L}_1 = \{$$
$$(1) \quad 4x - 2a \leq 1,$$
$$(2) \quad 7f(3b + 2) - 4x \leq -1,$$
$$(3) \quad 7f(3b + 2) - 2a \leq 0,$$
$$(6) \quad 2a - 7f(3b + 2) \leq 0,$$
$$(7) \quad 7f(3b + 2) - 3b \leq 1,$$
$$(9) \quad 3b - 2a \leq -1 \quad \}$$

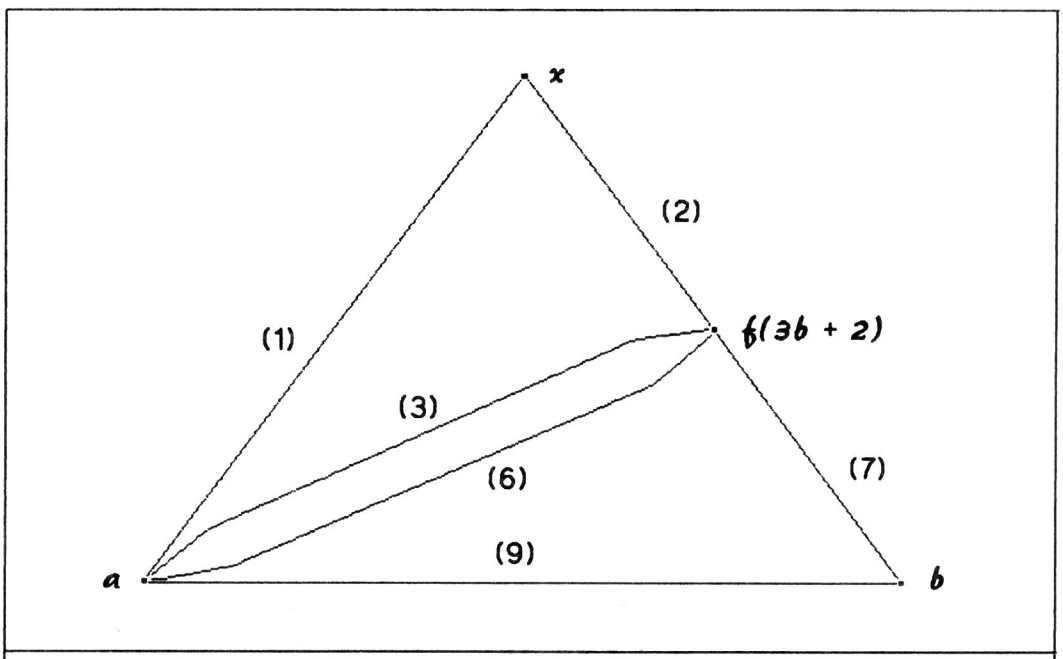


Figure 4.8.2.1: Graph $\mathcal{G}(\mathcal{L}_1)$

What has happened to the loops $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ of the graph $\mathcal{G}(\mathcal{L})$? Both loops $\mathcal{P}_2$ and $\mathcal{P}_4$ have lost an edge and $\mathcal{P}_3$ has changed a vertex. The reason is that the vertices of the loops $\mathcal{P}_2$ and $\mathcal{P}_3$ and the edges of the loops $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_4$ are not distinct. The loops $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_4$ all contain edge (5) of the graph $\mathcal{G}(\mathcal{L})$.
Out of the three equality loops, let us choose the modified loop $\mathcal{P}_3$ of the graph $\mathcal{G}(\mathcal{L}_1)$. We have lequalities

$$\mathcal{L}(\mathcal{P}_3) = \{ \quad (6) \quad 2a - 7\lfloor(3b + 2)\rfloor \le 0,$$
$$(7) \quad 7\lfloor(3b + 2)\rfloor - 3b \le 1,$$
$$(9) \quad 3b - 2a \le -1 \quad \}$$

and residue

$$r(\mathcal{P}_3) = \langle 2, -7, 0\rangle * \langle 7, -3, 1\rangle * \langle 3, -2, -1\rangle$$
$$= \langle 14, -21, 7\rangle * \langle 3, -2, -1\rangle$$
$$= \langle 42, -42, 0\rangle$$

with subresidues

$$\langle a_{11}, b_{11}, c_{11}\rangle = \langle 2, -7, 0\rangle$$
$$\langle a_{12}, b_{12}, c_{12}\rangle = \langle 14, -21, 7\rangle,$$

and thus

$$\mathcal{E}(\mathcal{P}_3) = \{ \quad 2a - 7\lfloor(3b + 2)\rfloor = 0,$$
$$14a - 21b = 7 \quad \}$$

with corresponding rewrite rules

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_3)) = \{ \lfloor(3b + 2)\rfloor \to (2/7)a,$$
$$b \to (2/3)a - (1/3) \ \}.$$

We therefore obtain the system $\mathcal{S}_2=(\mathcal{L}_2, \mathcal{E}_2, \phi)$ by the procedure LOOP_COLLABATION where

$$\mathcal{L}_2 = \{ \ (1) \quad 4x - 2a \le 1,$$
$$(2) \quad 2a - 4x \le -1 \ \}$$

and

$$\mathcal{E}_2 = \{ \ 2a - 7\lfloor(2a + 1)\rfloor = -1,$$
$$14a - 35y = -7,$$
$$2a - 7\lfloor(3b + 2)\rfloor = 0,$$
$$14a - 21b = 7 \ \}.$$

The lequality (3), $7f(3b + 2) - 2a \leq 0$, has been reduced to $0 \leq 0$ and therefore been dropped. The loop $\mathcal{P}_2$ has thus been erased completely. The loop $\mathcal{P}_4$ has again been reduced in its length. We simply obtain the system $\mathcal{S}_3 = (\emptyset, \mathcal{E}_3, \emptyset)$ by adding the equality $2a - 4x = -1$ to the set $\mathcal{E}_2$. And since there are no more lequalities left, the innermost REPEAT loop terminates. The procedure CANONIZATION is now called for the set

$$\begin{aligned}\mathcal{E}_3 = \{\ & 2a - 7f(2a + 1) = -1,\\ & 14a - 35y = -7,\\ & 2a - 7f(3b + 2) = 0,\\ & 14a - 21b = 7,\\ & 2a - 4x = -1\ \}.\end{aligned}$$

During the computation, this procedure reduces the constant b in the equality $2a - 7f(3b + 2) = 0$ by the rule $r(14a - 21b = 7)$ to the equality $2a - 7f(2a + 1) = 0$ which again is reduced by the rule $r(2a - 7f(2a + 1) = -1)$ to the equality $1 = 0$. It is thus returned that the system $\mathcal{S}$ is unsatisfiable.

Let us now consider how this result is obtained by chosing the central loop $\mathcal{P}_2$ of the graph $\mathcal{G}(\mathcal{L})$ as first equality loop:
The subresidues and equalities obtained by the equality loop $\mathcal{P}_2$ are

$$\begin{aligned}&\langle a_{11}, b_{11}, c_{11}\rangle = \langle 5, -7, 1\rangle\\ &\langle a_{12}, b_{12}, c_{12}\rangle = \langle 35, -49, 0\rangle,\\ &\mathcal{E}(\mathcal{P}_2) = \{\ 5y - 7f(3b + 2) = 1,\\ &\qquad\qquad 35y - 49f(2a + 1) = 0\ \},\end{aligned}$$

and the corresponding rewrite rules

$$\begin{aligned}\mathcal{R}(\mathcal{E}(\mathcal{P}_2)) = \{\ & f(3b + 2) \rightarrow (5/7)y - (1/7)\\ & f(2a + 1) \rightarrow (5/7)y\ \}.\end{aligned}$$

The procedure LOOP_COLLABATION produces the system $\mathcal{S}_1' = (\mathcal{L}_1', \mathcal{E}(\mathcal{P}_2), \emptyset)$ where the set $\mathcal{L}_1'$ is given by

$$\begin{aligned}\mathcal{L}_1' = \{\ &(1) && 4x - 5y \leq 0,\\ &(2) && 5y - 4x \leq 0,\\ &(4) && 2a - 5y \leq -1,\\ &(7) && 5y - 3b \leq 2,\\ &(8) && 5y - 2a \leq 1,\\ &(9) && 3b - 5y \leq -2\ \}\end{aligned}$$

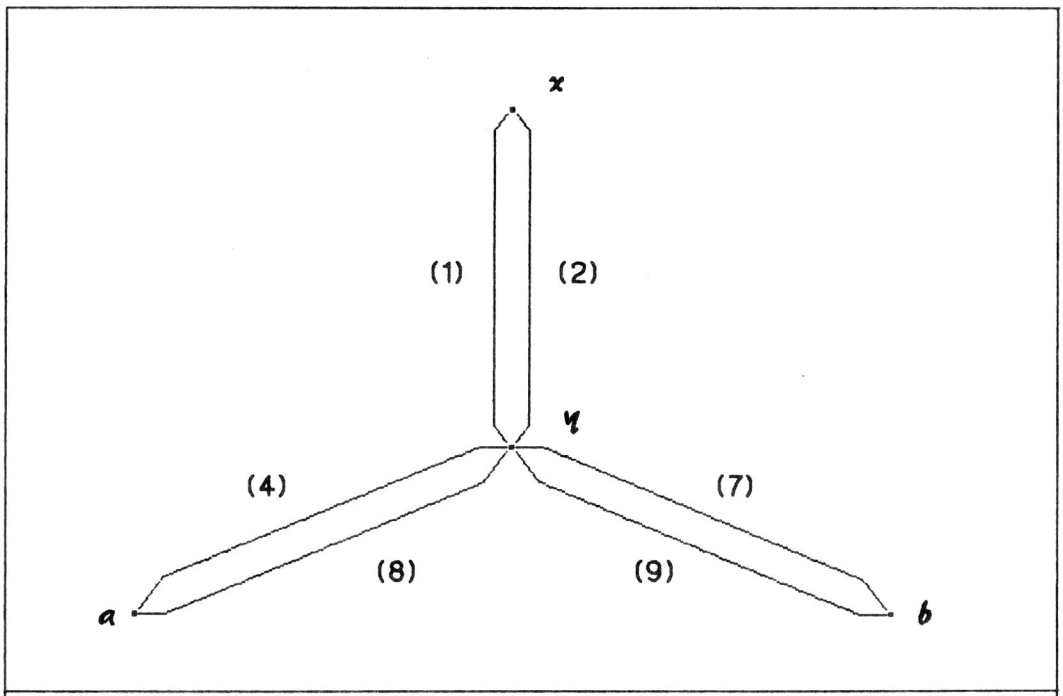


Figure 4.8.3.1: Graph $G(\mathcal{L}_1')$

The other equality loops $\mathcal{P}_1$, $\mathcal{P}_3$, and $\mathcal{P}_4$ have been reduced in their length. Let us now chose the modified loop $\mathcal{P}_1$ consisting of the edges (4) and (8). We thus have $\mathcal{E}(\mathcal{P}_1) = \{2a - 5y = -1\}$ and $\mathcal{S}_2' = (\mathcal{L}_2', \mathcal{E}(\mathcal{P}_1) \cup \mathcal{E}(\mathcal{P}_2), \phi)$ where

$$\mathcal{L}_2' = \{ \begin{array}{ll} (1) & 4x - 2a \leq 1, \\ (2) & 2a - 4x \leq -1, \\ (7) & 2a - 3b \leq 1, \\ (9) & 3b - 2a \leq -1 \ \}. \end{array}$$

The graph $G(\mathcal{L}_2')$ is given in figure 4.8.3.2.
Independent of the ordering in which the two simple equality loops of the graph $G(\mathcal{L}_2')$ are chosen, we obtain the system $\mathcal{S}_4' = (\phi, \mathcal{E}_4', \phi)$ with the set

$$\begin{aligned} \mathcal{E}_4' = \{ \quad & 5y - 7f(3b + 2) = 1, \\ & 35y - 49f(2a + 1) = 0, \\ & 2a - 5y = -1, \\ & 2a - 4x = -1, \\ & 2a - 3b = 1 \ \}. \end{aligned}$$

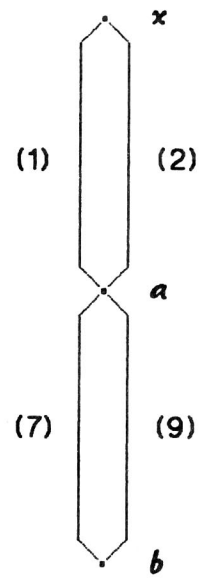


Figure 4.8.3.2: Graph $G(\mathcal{L}_2')$

Since the system $\mathcal{S}_4'$ has no lequalities, the innermost REPEAT loop terminates and the procedure CANONIZATION is called for the set $\mathcal{E}_4'$. Again, by reducing the constant b in the equality $5y - 7\mathfrak{f}(3b + 2) = 1$ by the rule $\mathfrak{r}(2a - 3b = 1)$ to the equality $5y - 7\mathfrak{f}(2a + 1) = 1$, and by reducing this equality by the rule $\mathfrak{r}(35y - 49\mathfrak{f}(2a + 1) = 0)$, we obtain the equality $1=0$. Therefore, the system $\mathcal{S}$ is unsatisfiable.

Let us finally consider how the unsatisfiability is detected if the loop $\mathcal{P}_4$ is chosen as first equality loop as argument of the procedure LOOP_COLLABATION.
The subresidues of the loop $\mathcal{P}_4$ are

$$\langle a_{11}, b_{11}, c_{11}\rangle = \langle 4, -7, 0\rangle$$
$$\langle a_{12}, b_{12}, c_{12}\rangle = \langle 28, -35, 0\rangle$$
$$\langle a_{13}, b_{13}, c_{13}\rangle = \langle 140, -245, 35\rangle$$

and therefore we have

$$\mathcal{E}(\mathcal{P}_4) = \{ \quad 4x - 7\mathfrak{f}(2a + 1) = 0,$$
$$28x - 35y = 0,$$
$$140x - 245\mathfrak{f}(3b + 2) = 35 \}$$

and

$$\mathcal{R}(\mathcal{E}(\mathcal{P}_4)) = \{\ f(2a+1) \to (4/7)x,\quad y \to (4/5)x,\quad f(3b+2) \to (4/7)x - (1/7)\ \}.$$

The procedure LOOP_COLLABATION constructs the system $\mathcal{S}_1'' = (\mathcal{L}_1'', \mathcal{E}(\mathcal{P}_4), \phi)$ where the set $\mathcal{L}_1''$ is given by

$$\begin{aligned}\mathcal{L}_1'' = \{\ &(4) && 2a - 4x \le -1,\\ &(7) && 4x - 3b \le 2,\\ &(8) && 4x - 2a \le 1,\\ &(9) && 3b - 4x \le -2 \qquad \}.\end{aligned}$$

Observe that all lequalities labeling edges of the loop $\mathcal{P}_2$ are removed since the lequalities (5) and (6) label edges of the loop $\mathcal{P}_4$ and since the lequality (3), $7f(3b+2) - 7f(2a+1) \le -1$, has been reduced to $0 \le 0$.

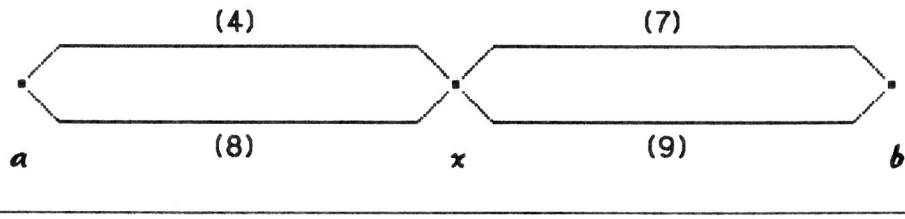


Figure 4.8.4.1: Graph $\mathcal{G}(\mathcal{L}_1'')$

Again, we have two simple equality loops in the graph $\mathcal{G}(\mathcal{L}_2'')$, and we obtain after two calls of the procedure LOOP_COLLABATION the system $\mathcal{S}_4'' = (\phi, \mathcal{E}_4'', \phi)$ where

$$\begin{aligned}\mathcal{E}_4'' = \{\ \ & 4x - 7f(2a+1) = 0,\\ & 28x - 35y = 0,\\ & 140x - 245f(3b+2) = 35,\\ & 2a - 4x = -1,\\ & 3b - 4x = -2 \qquad \}.\end{aligned}$$

The innermost REPEAT loop now terminates and the procedure CANONIZATION obtains from the last two equalities the equality $2a - 3b = 1$. We finally obtain the degenerated closed system $\mathcal{S}_5'' = (\phi, \{1=0\}, \phi)$ which is unsatisfiable for the Presburger Arithmetic.

4.8.2. Observations

The formula $\mathcal{F}_{fp}$ of the first example results in two systems without lequalities. They only consist of equalities and inequalities.
This formula has been used to demonstrate how the reduction process, described in figure 4.1, operates. Here, we want to use this example to point out the differences between the concept of rewriting and the congruence closure:
First of all, observe that the procedure CANONIZATION, a procedure computing a canonical ground rewriting system equivalent to a finite set of ground equations, contains one call of a congruence closure procedure according to section 4.4.2. Here, the congruence closure is constructed for the terms in that graph which represents only those terms occurring in the equalities.
On the other side, if a conjunction of equalities and inequalities is decided for satisfiability only by use of a congruence closure algorithm, the graph on which the congruence closure is computed, must represent each term occurring in the equalities *and inequalities*.
Therefore, if the systems of example 1 would be augmented by inequalities, the computational effort can significantly be increased for this method, while the computation of a canonical ground rewriting system for the set of equalities remains the same.
Roughly speaking, the congruence closure algorithm produces a maximal set of equalities, while a canonical rewriting system equivalent to a set of equations is minimal in the sense that no equation of a canonical set can be inferred by the other equations of that set.
The concept of rewriting thus serves for compactness in representing the given information.
Observe furthermore that lequalities can not be processed by a congruence closure algorithm. On the other side, rewrite rules can process terms wherever they occur.

It is underlined by the second example that the rewriting concepts are advantageously incorporated in the decision process of the validity of a formula of the quantifier-free Presburger Arithmetic extended by predicate and function symbols:
For the formula $\mathcal{F} \equiv (x \leq gx \wedge gx \leq x) \supset x = ggggggx$, the reduction process described in figure 4.1 would have produced a tremendous number of conjunctions to be decided for satisfiability in order to decide the validity of $\mathcal{F}$.

Shostak presented a procedure that decided the validity by constructing only seven conjunctions. He needs an ILP-solver for the ILP's associated with the conjunctions and the recursive procedure EQPAIRS; an explicit solution of a satisfiable ILP is required to determine the violations of well definedness by the procedure EQPAIRS.
By the EQUALITY_LOOP_RESIDUE procedure, presented in this diploma thesis, this computational effort reduces to the enumeration of one simple equality loop, the construction of one rewrite rule, and the normalization of one inequality. A fantastic improvement, isn't it?
Observe that the congruence closure algorithm alone is not directly applicable to $\mathcal{F}$ since it contains lequalities.

Example 3 points out that we have not yet succeeded in reducing the computational effort for deciding the validity of the given formula.
But that is no point for desperation. We present a solution for this problem in section 4.9.

The effects of the procedure LOOP_COLLABATION are particularly well demonstrated by the examples 4, 5, and 6. All systems initially consist only of lequalities.
In example 4, the simple equality loops encountered during the computation are all contained in the system given initially. Independently of their ordering, their processing results in an infeasible simple loop. Here, and in example 5, the unsatisfiability is based on the detection of an infeasible simple loop. In example 5, there exists only one simple admissible loop in any system of the constructed sequence. This example figuratively demonstrates the collabation of the graph.
Example 6 has especially been constructed to demonstrate how the collabation of a loop affects the other loops in the graph: Edges disappear and vertices are relabeled. And since the computation is not independent of the ordering, some different orderings have been examined. This example suggests to collabate the encountered equality loops according to their length in an non increasing order. The aspect of the ordering of the enumerated admissible simple loops is taken up in more detail in the next section 4.9.
For the example 6, the unsatisfiability is detected by reducing one of the obtained equalitities to $1=0$.

4.9. Further aspects and improvements

Since the generation of equalities by the procedure LOOP_COLLABATION plays a central role in the EQUALITY_LOOP_RESIDUE procedure, let us examine this procedure first.
The procedure LOOP_COLLABATION computes, according to its description, for a system $\mathcal{S}$ and an equality loop $\mathcal{P}$ the system $\mathcal{S}(\mathcal{P})$ whose equalities are obtained by augmenting those of $\mathcal{S}$ by $\mathcal{E}_{\mathcal{P}}$.
Since we have proved the equivalidity of the sets $\mathcal{L}_{\mathcal{P}=}$ and $\mathcal{E}$ for such a loop by Lemma 4.10, wouldn't it be easier to add the lequalities $\mathcal{L}_{\mathcal{P}\geq}$ to the set $\mathcal{L}$ or to replace $\mathcal{L}_{\mathcal{P}\leq}$ by $\mathcal{L}_{\mathcal{P}=}$ instead of replacing $\mathcal{L}_{\mathcal{P}\leq}$ by the set $\mathcal{E}_{\mathcal{P}}$ in the procedure LOOP_COLLABATION ?
Recall that the construction of the equalities of $\mathcal{E}_{\mathcal{P}}$ requires the computation of all subresidues of the loop $\mathcal{P}$ with a specific initial vertex, the vertex labeled with the smallest term t_1.
It would still be easier, but it would not be of advantage:
First of all, the set $\mathcal{L}_{\mathcal{P}=}$ is not minimal since the set $\mathcal{E}_{\mathcal{P}}$ is equivalent to $\mathcal{L}_{\mathcal{P}=}$ and has one equality less. Second, the right sides of the rewrite rules in $\mathcal{R}(\mathcal{E}_{\mathcal{P}})$ are already reduced wrt the rewrite rules $\mathcal{R}(\mathcal{E}_{\mathcal{P}})$ since only the β-term t_1 appears on the right side. And third, the terms $t_2, \ldots, t_n$ can be replaced in all lequalities and inequalities by using one application of a rewrite rule of $\mathcal{R}(\mathcal{E}_{\mathcal{P}})$ for each occurrence of t_i, $2 \leq i \leq n$. Finally, the graph for the set $\mathcal{L}(\mathcal{P})$ can easier be obtained from the graph $\mathcal{G}(\mathcal{L})$ if $\mathcal{E}_{\mathcal{P}}$ is used instead of $\mathcal{L}_{\mathcal{P}=}$.
All these are advantages of $\mathcal{E}_{\mathcal{P}}$. But what about the disadvantage of computing all subresidues of $\mathcal{P}$ subject to a specific initial vertex ?
This task can be transferred to the enumeration of the admissible simple loops in the procedure LOOP_RESIDUE, where the residue of each encountered admissible simple loop modulo cyclic permutation and reversal is computed by chosing that vertex labeled with the smallest term as initial vertex. The subresidues can be computed without any additional effort.
Let the set $\mathcal{E}_{\mathcal{C}}$ denote the set of equalities of the system before entrance into the innermost REPEAT loop and the sets $\mathcal{E}_1, \ldots, \mathcal{E}_n$ of equalities be obtained from the equality loops of one pass of the innermost REPEAT loop.

After termination of the innermost REPEAT loop, the procedure CANONIZATION is called for the set $\mathcal{E} = \mathcal{E}_C \cup \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_n$. Since the procedure CANONIZATION computes a canonical rewriting system equivalent to $\mathcal{E}$ according to the procedure given in figure 4.4, the congruence closure of $\mathcal{E}$ is constructed.
Observe that the procedure CONGRUENCE_CLOSURE of figure 3.3 incrementally computes the congruence closure of a set $\mathcal{R} \cup \{(u,v)\}$ from the congruence closure of $\mathcal{R}$ by calling the procedure MERGE with arguments u and v, and that the sets $\mathcal{E}_1, \ldots, \mathcal{E}_n$ correspond to subsets of the equivalence classes of the congruence closure of $\mathcal{E}$.
This observation can be used to reduce the computational effort for constructing the congruence closure of the set $\mathcal{E}$.
Further improvement is immediately possible in the case where $\mathcal{E}_C = \emptyset$ and all simple equality loops are initially contained in the graph. This is not an unusual case. Observe that examples 2, 3, 4, and 6 satisfy this condition.
In order to use the algorithm for computing a canonical ground rewriting system with more advantage, the enumeration of the simple admissible loops may be modified in such a way that the least terms of the enumerated loops form an increasing chain.
If the loops are enumerated in the suggested way, we can also incrementally compute a canonical rewriting system equivalent to $\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_n$ by adding the appropriate steps of the WHILE loop of figure 4.4 to the procedure LOOP_COLLABATION. We thus have a canonical rewriting system for $\mathcal{E}$ in case of $\mathcal{E}_C = \emptyset$.
Of course, if for $\mathcal{E}_C \neq \emptyset$ the set $\mathcal{E}_C$ could efficiently be modified and extended to a canonical system equivalent to $\mathcal{E}_C \cup \mathcal{E}_1 \cup \ldots \cup \mathcal{E}_n$, this improvement would be preferable to the previous suggestions.

The following example demonstrates that it may be of advantage to modify the procedure LOOP_COLLABATION in such a way that the set of equalities is augmented by a canonized set $C(\mathcal{E}_\mathcal{P})$ which is equivalent to $\mathcal{E}_\mathcal{P}$, instead of augmenting the equalities by $\mathcal{E}_\mathcal{P}$. The sets $\mathcal{L}(\mathcal{P})$ and $\mathcal{D}(\mathcal{P})$ are then advantageously obtained by reducing $\mathcal{L}$-$\mathcal{L}_\mathcal{P}$ and $\mathcal{D}$ wrt the rules $\mathcal{R}(C(\mathcal{E}_\mathcal{P})) \cup \mathcal{R}_{\mathcal{P}\mathcal{A}}$.
For example, consider the formula

$$\mathcal{F} \equiv \{ (f(a,c) \leq a \wedge a \leq f(a,c)) \wedge (x \leq g(x) \wedge g(x) \leq x) \wedge (b \leq h^3(b) \wedge h^3(b) \leq h^5(b) \wedge h^5(b) \leq b) \} \supset \{ f(f(a,c),c) = a \vee g^5(x) = x \vee h(b) = b \}.$$

Here, the hypothesis of the formula $\mathcal{F}$ is a conjunction, and the negation of this formula,

$$\neg\mathcal{F} \equiv \{ (f(a,c)\le a \wedge a\le f(a,c)) \wedge (x\le g(x) \wedge g(x)\le x) \wedge (b\le h^3(b) \wedge h^3(b)\le h^5(b) \wedge h^5(b)\le b) \} \wedge \{ f(f(a,c),c)\neq a \wedge g^5(x)\neq x \wedge h(b)\neq b \}$$

results in one system $\mathcal{S}=(\mathcal{L},\phi,\mathcal{D})$ for its disjunctive normal form where

$$\mathcal{L} = \{ f(a,c)\le a,\ a\le f(a,c)),\ x\le g(x),\ g(x)\le x),\ b\le h^3(b),\ h^3(b)\le h^5(b),\ h^5(b)\le b \},$$
$$\mathcal{D} = \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b \}).$$

The EQUALITY_LOOP_RESIDUE procedure may construct a sequence of two systems corresponding to the encountered equality loops. For example,

$$\mathcal{S}_1 = (\{ f(a,c)\le a,\ a\le f(a,c)),\ x\le g(x),\ g(x)\le x) \}, \{ b=h^3(b),\ b=h^5(b) \}, \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b \}),$$

and $\mathcal{S}_2 = (\{ x\le g(x),\ g(x)\le x) \}, \{ b=h^3(b),\ b=h^5(b),\ f(a,c)=a \}, \{ 0\neq 0,\ g^5(x)\neq x,\ h(b)\neq b \}).$

But the suggested version would canonize the set $\{b=h^3(b),\ b=h^5(b)\}$. Only

$$\mathcal{S}_1' = (\{ f(a,c)\le a,\ a\le f(a,c)),\ x\le g(x),\ g(x)\le x) \}, \{ b=h(b) \}, \{ f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ 0\neq 0 \})$$

would be constructed to detect the unsatisfiabilty.

Since ground rewrite rules can always be directed and since a Noetherian system is confluent if it is locally confluent, the set $\mathcal{E}_\mathcal{P}$ for an equality loop $\mathcal{P}$ is a canonical system unless there is a critical pair consisting of two terms which label vertices of the loop such that one term is a subterm of the other. In the example, $h^3(b)$ is a subterm of $h^5(b)$ and both terms label vertices of the same equality loop.

If either of the other two simple equality loops would have been encountered, the procedure LOOP_COLLABATION would have immediately obtained the inequality $0\neq 0$. This is not an unusual case since many formulas in program verification are of a simple form. The canonization of the set $\mathcal{E}_\mathcal{P}$ thus requires only little additional effort and the unsatisfiability may be detected much earlier.

Example 3 was deliberately constructed from three formulas to make the multiplicative effect on the number of conjunctions in the disjunctive normal form obvious.
Since the expansion into disjunctive normal form of the negated formula $\neg\mathcal{F}$ results in 24 conjunctions which groupwise contain same information (i.e., $f(a,c)\leq a$ and $a\leq f(a,c)$ occurs in 8 conjunctions), the compactness of the presentation of the information must have been lost by the expansion process.
This gives rise to a new idea: Instead of omitting the chance to work on the original formula, we suggest to compact the formula in a perspicacious way described in figure 4.9 .
A formula C is a subconjunction of a formula $\mathcal{F}$ if C is a conjunction and a subformula of $\mathcal{F}$. It is obvious that any replacement of C by an equivalent formula does not change the validity. It suffices to take only maximal subconjunctions into consideration. Otherwise, redundant computation is done.
Let us now turn our attention back to example 3:

$$\begin{aligned}\mathcal{F} \equiv \quad & \{\ (f(a,c)\leq a \wedge a\leq f(a,c)) \quad \vee \\ & (x\leq g(x) \wedge g(x)\leq x) \quad \vee \\ & (\ b\leq h^3(b) \wedge h^3(b)\leq h^5(b) \wedge h^5(b)\leq b\ )\ \} \\ & \supset \\ & \{\ (f(f(a,c),c)\leq a \wedge a\leq f(f(a,c),c)) \quad \vee \\ & (g^5(x)\leq x \wedge x\leq g^5(x)) \quad \vee \\ & (h(b)\leq b \wedge b\leq h(b))\ \}.\end{aligned}$$

Before it is negated and expanded into disjunctive normal form, the procedure described in figure 4.9 transforms this formula to the equivalent formula

$$\begin{aligned}& \{\ a=f(a,c) \quad \vee \quad x=g(x) \quad \vee \quad b=h(b)\ \} \\ & \supset \\ & \{\ f(f(a,c),c)=a \quad \vee \quad g^5(x)=x \quad \vee \quad h(b)=b\ \}\end{aligned}$$

Its negation is now expanded and consists of only three systems, namely,

$$S_1 = (\phi,\ \{\ a=f(a,c)\ \},\ \{f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b\}\ )$$

$$S_2 = (\phi,\ \{\ x=g(x)\ \},\ \{f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b\}\ )\quad \text{and}$$

$$S_3 = (\phi,\ \{\ b=h(b)\ \},\ \{f(f(a,c),c)\neq a,\ g^5(x)\neq x,\ h(b)\neq b\}\ )$$

Each system is immediately proved as unsatisfiable by one call to the procedures CANONIZATION and REDUCTION. We thus have an eightfold reduction in the number of constructed systems.

FOR *each subconjunction* C *of* $\mathcal{F}$
DO BEGIN
    apply the EQUALITY_LOOP_RESIDUE *procedure*
    to the corresponding system $S(C)$ *of* C
    IF $S(C)$ *is satisfiable*
      THEN *let* $C(S)$ *be the conjunction that corresponds to the obtained closed system*
      ELSE *let* $C(S)$ *be* $0 \neq 0$
    replace in $\mathcal{F}$ *the subconjunction* C *by* $C(S)$
  END
Let $\mathcal{F}_N$ *be the formula obtained from* $\neg\mathcal{F}$ *by moving the negation inwards, i.e., replacing* $\neg(A \vee B)$ *by* $\neg A \wedge \neg B$.
FOR *each subconjunction* C *of* $\mathcal{F}_N$
DO BEGIN
    apply the EQUALITY_LOOP_RESIDUE *procedure*
    to the corresponding system $S(C)$ *of* C
    IF $S(C)$ *is satisfiable*
      THEN *let* $C(S)$ *be the conjunction that corresponds to the obtained closed system*
      ELSE *let* $C(S)$ *be* $0 \neq 0$
    replace in $\mathcal{F}_N$ *the subconjunction* C *by* $C(S)$
  END
Expand the resulting formula into disjunctive normal form, obtaining $\mathcal{F}_{ND}$, *and construct a system* S *for each conjunction* C *of the disjunctive normal form* $\mathcal{F}_{ND}$.
FOR *each system* S
DO BEGIN
    apply the EQUALITY_LOOP_RESIDUE
    procedure to system S
    IF S *is satisfiable*
      THEN RETURN ($\mathcal{F}$ is not valid)
  END
RETURN ($\mathcal{F}$ is valid)

Figure 4.9: Algorithm deciding the validity of a formula $\mathcal{F} \in PA_6$

5. Conclusion

This diploma thesis presents a new decision procedure for the quantifier-free Presburger Arithmetic extended by predicate and function symbols, the EQUALITY LOOP RESIDUE procedure.
The well definedness of function symbols serves for combinatorial explosion in the complexity of this decision problem if a sledgehammer is used like in the procedure of the decidability proof in section 4.1.
Shostak has postponed this problem and first searches for a solution of the associated ILP. If one is found and a violation is detected, he explicitely computes a formula that summarizes the violation.
On the other side, the EQUALITY LOOP RESIDUE procedure can implicitely guarantee the well definedness of the function symbols. This leads to a further milestone in reducing the computational effort of this decision problem which is essential to mechanical theorem proving, program verification, and other tasks.
Basically, it is a combination of two concepts: The concept of the LOOP_RESIDUE method, originally designed for the unextended class, and the concept of rewriting on ground terms which makes use of the information which is compactly represented by additional use of = and ≠.
As it is evident by the examples in section 4.8, the computational effort can significantly be reduced.
Though the worst case time complexity of the EQUALITY LOOP RESIDUE PROCEDURE is still exponential since it uses a modified version of the Loop-Residue procedure as subroutine, the examples have pointed out that there is still enough room left for an improvement of such decision procedures.
Furthermore, we have seen in section 4.9 that the application of this procedure is not restriced to the quantifier-free conjunctions of a disjunctive normal form:
As shown by the example

$$\begin{aligned}\mathcal{F} \equiv \quad & \{\ (f_f(a,c) \le a \wedge a \le f_f(a,c)) \quad \vee \\ & (x \le g(x) \wedge g(x) \le x) \quad \vee \\ & (\ b \le h^3(b) \wedge h^3(b) \le h^5(b) \wedge h^5(b) \le b\ )\ \} \\ & \supset \\ & \{\ (f_f(f_f(a,c),c) \le a \wedge a \le f_f(f_f(a,c),c)) \quad \vee \\ & (g^5(x) \le x \wedge x \le g^5(x)) \quad \vee \\ & (h(b) \le b \wedge b \le h(b))\ \},\end{aligned}$$

the most effective way to use the new ideas is to disassociate with the conventional method of negating and expanding into disjunctive normal form and with the self-made restriction of concentrating only on the decision algorithm for the satisfiability of a system.
On the contrary to the conventional method, this example indicates that information should be compacted and processed, even before the negation is expanded into disjunctive normal form and even before the formula is negated.
Some interesting aspects have been pointed out in section 4.9 and a lot of suggestions for improvements have been made which have to be evaluated in detail. According to the examples and the suggested improvements, it quite seems that the new procedure, the EQUALITY LOOP RESIDUE procedure, is only the first step of a substantial improvement of decision procedures of the quantifier-free Presburger Arithmetic extended by predicate and function symbols.

Appendix A:
Proof of the main theorem for the unextended class

Since theorem 2.3 is fundamental for the procedure of the extended class, its proof is given here in detail. It is a refined version of the proof presented in [24] in which for some cases the idea is given and the proof was left to the reader.

Lemma A.1:
The concatenation $\mathcal{PQ}$ of two admissible paths $\mathcal{P}$ and $\mathcal{Q}$ is admissible if and only if $sgn(b_{\mathcal{P}} * a_{\mathcal{Q}}) = -1$

Proof:
Note that $sgn(a) = sgn(sgn(a_2)a_1a_2) = sgn(a_2a_1a_2) = sgn(a_1)$
and $sgn(b) = sgn(-sgn(a_2)b_1b_2) = sgn(b_1b_1b_2) = sgn(b_2)$
for $sgn(b_1a_2) = -1$ and $\langle a,b,c\rangle = \langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle$.
And, by induction, $sgn(a)=sgn(a_1)$ and $sgn(b)=sgn(b_n)$
for $\langle a,b,c\rangle = \langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle * \ldots * \langle a_n,b_n,c_n\rangle$
with $sgn(b_ia_{i+1}) = -1$ for $1 \le i < n$.
Recalling the definition of admissibility, the lemma thus follows. □

Corollary A.2:
For an admissible loop $\mathcal{P}$ and its reverse $\mathcal{Q}$, we have
i) $\mathcal{P}$ is infeasible if and only if $\mathcal{Q}$ is infeasible
ii) $\mathcal{P}$ is an equality loop if and only if $\mathcal{Q}$ is an equality loop

Proof:
If we denote by the function $^\sim$ the change $\langle a,b,c\rangle^\sim := \langle b,a,c\rangle$ on triples, then

$$\begin{aligned}
&\langle a_2,b_2,c_2\rangle^\sim * \langle a_1,b_1,c_1\rangle^\sim \\
&= \langle b_2,a_2,c_2\rangle * \langle b_1,a_1,c_1\rangle \\
&= sgn(b_1)\,\langle b_2b_1,\ -a_2a_1,\ (c_2b_1 - c_1a_2)\rangle \\
&= sgn(a_2)\,\langle -b_1b_2,\ a_1a_2,\ (c_1a_2 - c_2b_1)\rangle \\
&= (\langle a_1,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle)^\sim
\end{aligned}$$

and by induction

$$*_{i=0}^{n}\,(\langle a_{n-i},b_{n-i},c_{n-i}\rangle^\sim) = (*_{i=0}^{n}\,\langle a_i,b_i,c_i\rangle)^\sim.$$

Thus,

$$\begin{aligned}
\tau(\mathcal{P})^\sim &= \langle a(\mathcal{P}),b(\mathcal{P}),c(\mathcal{P})\rangle^\sim = \langle b(\mathcal{P}),a(\mathcal{P}),c(\mathcal{P})\rangle \\
&= \langle a(\mathcal{Q}),b(\mathcal{Q}),c(\mathcal{Q})\rangle = \tau(\mathcal{Q}).
\end{aligned}$$

i) $\mathcal{P}$ is infeasible
iff ($a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) < 0$)
iff ($a(\mathcal{Q}) + b(\mathcal{Q}) = 0$ and $c(\mathcal{Q}) < 0$)
iff $\mathcal{Q}$ is infeasible

ii) $\mathcal{P}$ is an equality loop
iff ($a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) = 0$)
iff ($a(\mathcal{Q}) + b(\mathcal{Q}) = 0$ and $c(\mathcal{Q}) = 0$)
iff $\mathcal{Q}$ is an equality loop □

Corollary A.3:
For an admissible permutable loop $\mathcal{P}$ and a cyclic permutation $\mathcal{P}'$ of $\mathcal{P}$,
i) $\mathcal{P}$ is infeasible if and only if $\mathcal{P}'$ is infeasible
ii) $\mathcal{P}$ is an equality loop if and only if $\mathcal{P}'$ is an equality loop

Proof:
If $\mathcal{P}=\mathcal{P}'$, we are done. Otherwise, there are paths $\mathcal{Q}$ and $\mathcal{R}$ with $\mathcal{P}=\mathcal{QR}$, $\mathcal{P}'=\mathcal{RQ}$,

$$\begin{aligned} v(\mathcal{P}) &= \langle a(\mathcal{P}), b(\mathcal{P}), c(\mathcal{P})\rangle \\ &= sgn(a(\mathcal{R}))\ \langle a(\mathcal{Q})a(\mathcal{R}), -b(\mathcal{Q})b(\mathcal{R}), (c(\mathcal{Q})a(\mathcal{R}) - c(\mathcal{R})b(\mathcal{Q}))\rangle, \\ v(\mathcal{P}') &= \langle a(\mathcal{P}'), b(\mathcal{P}'), c(\mathcal{P}')\rangle \\ &= sgn(a(\mathcal{Q}))\ \langle a(\mathcal{R})a(\mathcal{Q}), -b(\mathcal{R})b(\mathcal{Q}), (c(\mathcal{R})a(\mathcal{Q}) - c(\mathcal{Q})b(\mathcal{R}))\rangle, \end{aligned}$$

and, by Lemma A.1, $sgn(b(\mathcal{R})a(\mathcal{Q})) = sgn(b(\mathcal{Q})a(\mathcal{R})) = -1$.

Denote by $<^{\omega}$ for $\omega \neq 0$ the relation $A <^{\omega} B$ iff $A\omega < B\omega$ and analogously by $\leq^{\omega}$ for $\omega \neq 0$ the relation $A \leq^{\omega} B$ iff $A\omega \leq B\omega$.
Now,

i) $\mathcal{P}$ is infeasible
iff $a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) < 0$
iff $a(\mathcal{Q})a(\mathcal{R}) - b(\mathcal{Q})b(\mathcal{R}) = 0$ and $c(\mathcal{Q})a(\mathcal{R}) <^{a(\mathcal{R})} c(\mathcal{R})b(\mathcal{Q})$
iff $b(\mathcal{R})/a(\mathcal{R}) = a(\mathcal{Q})/b(\mathcal{Q})$ and $c(\mathcal{Q})a(\mathcal{R}) <^{a(\mathcal{R})} c(\mathcal{R})b(\mathcal{Q})$
iff $b(\mathcal{R})/a(\mathcal{R}) = a(\mathcal{Q})/b(\mathcal{Q})$ and $c(\mathcal{Q})b(\mathcal{R}) <^{b(\mathcal{R})} c(\mathcal{R})a(\mathcal{Q})$ [29]
iff $a(\mathcal{R})a(\mathcal{Q}) - b(\mathcal{R})b(\mathcal{Q}) = 0$ and $c(\mathcal{R})a(\mathcal{Q}) <^{a(\mathcal{Q})} c(\mathcal{Q})b(\mathcal{R})$
(recalling $sgn(b(\mathcal{R})a(\mathcal{Q})) = -1$)
iff $a(\mathcal{P}') + b(\mathcal{P}') = 0$ and $c(\mathcal{P}') < 0$
iff $\mathcal{P}'$ is infeasible.

29 : Observe that $Ax <^{\omega x} Bx$ iff $A <^{\omega} B$ and $x \neq 0$.

In the same way, we have

ii) $\mathcal{P}$ is an equality loop

iff $a(\mathcal{P}) + b(\mathcal{P}) = 0$ and $c(\mathcal{P}) = 0$

iff $a(\mathcal{Q})a(\mathcal{R}) - b(\mathcal{Q})b(\mathcal{R}) = 0$ and $c(\mathcal{Q})a(\mathcal{R}) = c(\mathcal{R})b(\mathcal{Q})$

iff $b(\mathcal{R})/a(\mathcal{R}) = a(\mathcal{Q})/b(\mathcal{Q})$ and $c(\mathcal{Q})a(\mathcal{R}) = c(\mathcal{R})b(\mathcal{Q})$

iff $b(\mathcal{R})/a(\mathcal{R}) = a(\mathcal{Q})/b(\mathcal{Q})$ and $c(\mathcal{Q})b(\mathcal{R}) = c(\mathcal{R})a(\mathcal{Q})$

iff $a(\mathcal{R})a(\mathcal{Q}) - b(\mathcal{R})b(\mathcal{Q}) = 0$ and $c(\mathcal{R})a(\mathcal{Q}) = c(\mathcal{Q})b(\mathcal{R})$

(recalling $sgn(b(\mathcal{R})a(\mathcal{Q})) = -1$)

iff $a(\mathcal{P}') + b(\mathcal{P}') = 0$ and $c(\mathcal{P}') = 0$

iff $\mathcal{P}'$ is an equality loop. □

Corollary A.2.i) and A.3.i) refer to the definition of the closure of a graph: Since a closure has for each admissible simple loop modulo cyclic permutation and reversal an edge labeled with its residue lequality, the closure of a graph is not uniquely defined. But for two closures G_1 and G_2, G_1 has an admissible simple loop if and only if G_2 has an admissible simple loop by the corollaries.

Definition: Define the discriminant $\partial_{\mathcal{P}}$, or $\partial(\mathcal{P})$, of an admissible path $\mathcal{P}$ by $\partial_{\mathcal{P}} = c_{\mathcal{P}}/(a_{\mathcal{P}}+b_{\mathcal{P}})$, where $\langle a_{\mathcal{P}}, b_{\mathcal{P}}, c_{\mathcal{P}} \rangle$ is the residue of $\mathcal{P}$.

Lemma A.4:

If $\mathcal{PQ}$ is an admissible loop from v_0 to v_0, then the following statements are equivalent:

i) $\mathcal{PQ}$ is infeasible

ii) $\partial_{\mathcal{P}} > \partial_{\mathcal{Q}}$ (with $a_{\mathcal{Q}}$)

iii) $\partial_{\mathcal{P}} < \partial_{\mathcal{Q}}$ (with $b_{\mathcal{P}}$)

Proof:

If $\mathcal{PQ}$ is an admissible loop from v_0 to v_0, then $a(\mathcal{P}) = b(\mathcal{P}) = 0$, and by Lemma 1, $sgn(-b(\mathcal{P})a(\mathcal{Q})) = 1$. Obviously, ii) and iii) are equivalent.

We have

$$r(\mathcal{PQ}) = r(\mathcal{P}) * r(\mathcal{Q}) = \langle 0, b(\mathcal{P}), c(\mathcal{P}) \rangle * \langle a(\mathcal{Q}), 0, c(\mathcal{Q}) \rangle$$
$$= \langle 0, 0, sgn(a(\mathcal{Q}))\,(c(\mathcal{P})a(\mathcal{Q}) - c(\mathcal{Q})b(\mathcal{P})) \rangle$$

Thus,

$\mathcal{PQ}$ is infeasible

iff $c(\mathcal{P})a(\mathcal{Q}) < a(\mathcal{Q})\ c(\mathcal{Q})b(\mathcal{P})$ [$\because (-b(\mathcal{P})a(\mathcal{Q}))$]

iff $-c(\mathcal{P})/b(\mathcal{P}) < a(\mathcal{Q})\ -c(\mathcal{Q})/a(\mathcal{Q})$

iff $c(\mathcal{P})/b(\mathcal{P}) > a(\mathcal{Q})\ c(\mathcal{Q})/a(\mathcal{Q})$

iff $\partial(\mathcal{P}) > a(\mathcal{Q})\ \partial(\mathcal{Q})$ □

Lemma A.5:
If a closed graph $\mathcal{G}$ has an infeasible loop from v_0 to v_0, then $\mathcal{G}$ has an infeasible simple loop.

Proof:
Let $\mathcal{P}$ be a shortest infeasible loop from v_0 to v_0 in $\mathcal{G}$. If $\mathcal{P}$ is simple, we are done. Otherwise, $\mathcal{P}$ can be expressed as $\mathcal{P}_1\mathcal{P}_2\mathcal{P}_3$, where $\mathcal{P}_2$ is an admissible simple loop. We thus have

(1) $a_1=0$ and $b_3=0$ (v_0 is the first and last vertex of $\mathcal{P}$)

(2) $sgn(b_1a_2) = sgn(b_2a_3) = -1$ (by Lemma A.1)

$$r(\mathcal{P}_1\mathcal{P}_2) = \langle 0,b_1,c_1\rangle * \langle a_2,b_2,c_2\rangle = sgn(a_2)\langle 0,-b_1b_2,(c_1a_2-c_2b_1)\rangle$$

$$(3)\ \partial(\mathcal{P}_1\mathcal{P}_2) = \frac{c_1a_2-c_2b_1}{-b_1b_2} = -\frac{a_2}{b_2}*\frac{c_1}{b_1}+\frac{c_2}{b_2} = -\frac{a_2}{b_2}*\partial(\mathcal{P}_1)+\frac{c_2}{b_2}$$

$$r(\mathcal{P}_2\mathcal{P}_3) = \langle a_2,b_2,c_2\rangle * \langle a_3,0,c_3\rangle = sgn(a_3)\langle a_2a_3,0,c_2a_3-c_3b_2\rangle$$

$$(4)\ \partial(\mathcal{P}_2\mathcal{P}_3) = \frac{c_2a_3-c_3b_2}{a_2a_3} = \frac{c_2}{a_2} - \frac{b_2}{a_2}*\frac{c_3}{a_3} = \frac{c_2}{a_2} - \frac{b_2}{a_2}*\partial(\mathcal{P}_3)$$

(5) $\partial(\mathcal{P}_1\mathcal{P}_2) >^{a_3} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{P}_1) >^{a_2} \partial(\mathcal{P}_2\mathcal{P}_3)$
(by Lemma A.4 and the infeasibility of $\mathcal{P}$).

We now proof by contradiction that the simple admissible loop $\mathcal{P}_2$ is infeasible. Suppose $\mathcal{P}_2$ is not infeasible. Then either I) $a_2+b_2=0$ and $c_2\geq 0$ or II) $a_2+b_2\neq 0$.

Because of the length of the proof, the following survey of the distinguished cases may be very helpful to reconsider the proof:

I $a_2+b_2=0$ and $c_2\geq 0$.
II $a_2+b_2\neq 0$.
- II. A $\mathcal{P}_2$ is not permutable
- II. B $\mathcal{P}_2$ is permutable and $\mathcal{P}_2=\mathcal{P}_2'$
  - II.B. 1 $sgn(a)=sgn(a_2)$
  - II.B.2 $sgn(a)=sgn(b_2)$
- II. C $\mathcal{P}_2$ is permutable and $\mathcal{P}_2\neq\mathcal{P}_2'$
  - II.C.1 $sgn(a_2+b_2)=sgn(a_2)$
  - II.C.2 $sgn(a_2+b_2)=sgn(b_2)$.

I) Assume $a_2+b_2=0$ and $c_2 \geq 0$.
Since by Lemma A.1, (2), and $b_2 = -a_2$,

$$sgn(b_1a_3) = sgn(b_1(b_1a_2b_2a_3)a_3)$$
$$= sgn(a_2b_2) = sgn(-a_2a_2) = -1.$$

Therefore, the loop $\mathcal{P}_1\mathcal{P}_3$ is admissible. Since $sgn(a_2a_3)=sgn(-b_2a_3)=1$, we have

$$-c_2/b_2 = c_2/a_2 \geq^{a_3} 0.$$

So by (3), (5), and Lemma A.4,

$$\partial(\mathcal{P}_1) = (-b_2/a_2)\partial(\mathcal{P}_1) = \partial(\mathcal{P}_1\mathcal{P}_2) - c_2/b_2$$
$$\geq^{a_3} \partial(\mathcal{P}_1\mathcal{P}_2) >^{a_3} \partial(\mathcal{P}_3).$$

The loop $\mathcal{P}_1\mathcal{P}_3$ is thus an infeasible loop from v_0 to v_0 and we have a contradiction to the assumption that $\mathcal{P}=\mathcal{P}_1\mathcal{P}_2\mathcal{P}_3$ is a shortest infeasible loop from v_0 to v_0.

II) Assume $a_2+b_2 \neq 0$.
Now the closedness of $\mathcal{G}$ provides an edge $\mathcal{E}$ labeled with $ax \leq c$ that connects some vertex x of the admissible loop $\mathcal{P}_2$ with the vertex v_0, where c/a is the discriminant of some cyclic permutation $\mathcal{P}_2'$ of $\mathcal{P}_2$.
We have the following three cases:

A) $\mathcal{P}_2$ is not permutable,
B) $\mathcal{P}_2$ is permutable and $\mathcal{P}_2=\mathcal{P}_2'$, and
C) $\mathcal{P}_2$ is permutable and $\mathcal{P}_2 \neq \mathcal{P}_2'$.

II.A) $\mathcal{P}_2$ is not permutable:
We then have $sgn(a_2b_2)=1$ by Lemma A.1.
Thus,

$$\mathcal{P}_2=\mathcal{P}_2',\ a=a_2+b_2,\ c=c_2,$$

and

$$sgn(a) = sgn(a_2+b_2) = sgn(a_2) = sgn(b_2).$$

Since

$$sgn(b_1a)=sgn(b_1a_2)=-1$$

and $sgn(aa_3)=sgn(b_2a_3)=-1$,

both $\mathcal{P}_1\mathcal{E}$ and $\mathcal{E}\mathcal{P}_3$ are admissible loops from v_0 to v_0.
By (3) and (4), we have

$$\frac{c_2}{b_2} = \partial(\mathcal{P}_1\mathcal{P}_2) + \frac{a_2}{b_2} * \partial(\mathcal{P}_1) \qquad \frac{c_2}{a_2} = \partial(\mathcal{P}_2\mathcal{P}_3) + \frac{b_2}{a_2} * \partial(\mathcal{P}_3)$$

and since (5), and $sgn(a)=sgn(a_2)=-sgn(a_3)$:

$$\partial(\mathcal{P}_1\mathcal{P}_2) <^a \partial(\mathcal{P}_3)$$

and $\partial(\mathcal{P}_2\mathcal{P}_3) <^a \partial(\mathcal{P}_1)$.

We claim that at least one of $\mathcal{P}_1\mathcal{E}$ and $\mathcal{E}\mathcal{P}_3$ is infeasible. Otherwise, by Lemma A.4, $\partial(\mathcal{P}_1) \leq^a \partial(\mathcal{E})$ and $\partial(\mathcal{E}) \leq^{a_3} \partial(\mathcal{P}_3)$ (or, since $sgn(aa_3)=-1$, $\partial(\mathcal{P}_3) \leq^a \partial(\mathcal{E})$). Thus,

$$\begin{aligned}
&2\partial(\mathcal{E}) \\
&=2\partial(\mathcal{P}_2) \\
&= \frac{c_2}{b_2}\left(\frac{b_2}{a_2+b_2}\right) + \frac{c_2}{a_2}\left(\frac{a_2}{a_2+b_2}\right) \\
&= \left(\frac{a_2}{b_2}\partial(\mathcal{P}_1) + \partial(\mathcal{P}_1\mathcal{P}_2)\right)\left(\frac{b_2}{a_2+b_2}\right) + \left(\frac{b_2}{a_2}\partial(\mathcal{P}_3) + \partial(\mathcal{P}_2\mathcal{P}_3)\right)\left(\frac{a_2}{a_2+b_2}\right) \\
&<^a \left(\frac{a_2}{b_2}\partial(\mathcal{P}_1) + \partial(\mathcal{P}_3)\right)\left(\frac{b_2}{a_2+b_2}\right) + \left(\frac{b_2}{a_2}\partial(\mathcal{P}_3) + \partial(\mathcal{P}_1)\right)\left(\frac{a_2}{a_2+b_2}\right) \\
&\leq^a \left(\frac{a_2}{b_2}\partial(\mathcal{E}) + \partial(\mathcal{E})\right)\left(\frac{b_2}{a_2+b_2}\right) + \left(\frac{b_2}{a_2}\partial(\mathcal{E}) + \partial(\mathcal{E})\right)\left(\frac{a_2}{a_2+b_2}\right) \\
&= 2\partial\mathcal{E},
\end{aligned}$$

a contradiction. Therefore, either $\mathcal{P}_1\mathcal{E}$ or $\mathcal{E}\mathcal{P}_3$ is an infeasible loop from v_0 to v_0 contradicting the shortness of $\mathcal{P}$.

II.B) $\mathcal{P}_2$ is permutable and $\mathcal{P}_2=\mathcal{P}_2^*$.

We then have $sgn(a_2b_2)=-1$ by Lemma A.1. Recall that $a=a_2+b_2$ and

$$\begin{aligned}
&sgn(b_1a_3) \\
&=sgn(b_1(a_2a_2)(b_2b_2)a_3) \\
&=sgn(b_1a_2)sgn(a_2b_2)sgn(b_2a_3) \\
&=(-1)*(-1)*(-1) \\
&=-1.
\end{aligned}$$

We distinguish between the two cases

1) $sgn(a)=sgn(a_2)$ and
2) $sgn(a)=sgn(b_2)$.

II.B.1) $sgn(a)=sgn(a_2)$.
Since

$$sgn(b_1a_3) = -1$$

and

$$sgn(b_1a) = sgn(b_1a_2) = -1,$$

both $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{P}_1\mathcal{E}$ are, by Lemma A.1, admissible loops.
Suppose neither $\mathcal{P}_1\mathcal{P}_3$ nor $\mathcal{P}_1\mathcal{E}$ is infeasible.
Then, by lemma A.4, $\partial(\mathcal{P}_1) \leq^{a_3} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{P}_1) \leq^{a} \partial(\mathcal{E})$ (or equivalenty $\partial(\mathcal{P}_1) \leq^{a_2} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{P}_1) \leq^{a_2} \partial(\mathcal{P}_2)$, since $\partial(\mathcal{P}_2)=\partial(\mathcal{E})$ and $sgn(a_2a_3)=sgn(-b_2a_3)=1$). Thus by (5)

$$\begin{aligned}
\partial(\mathcal{P}_1) &= \left(\frac{a_2+b_2}{a_2}\right)\partial(\mathcal{P}_1) - \frac{b_2}{a_2}\partial(\mathcal{P}_1) \\
&\leq^{a_2} \left(\frac{a_2+b_2}{a_2}\right)\partial(\mathcal{P}_2) - \frac{b_2}{a_2}\partial(\mathcal{P}_3) \\
&= \frac{c_2}{a_2} - \frac{b_2}{a_2}\partial(\mathcal{P}_3) \\
&= \partial(\mathcal{P}_2\mathcal{P}_3) <^{a_2} \partial(\mathcal{P}_1),
\end{aligned}$$

a contradiction.
Therefore, at least one of $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{P}_1\mathcal{E}$ is infeasible. But this once more contradicts the shortness of $\mathcal{P}$.

II.B.2) $sgn(a)=sgn(b_2)$.
Since

$$sgn(b_1a_3) = -1$$

and

$$sgn(aa_3) = sgn(b_2a_3) = -1$$

both $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{E}\mathcal{P}_3$ are, by Lemma A.1, admissible loops.
Suppose neither $\mathcal{P}_1\mathcal{P}_3$ nor $\mathcal{E}\mathcal{P}_3$ is infeasible.
Then, by lemma A.4, $\partial(\mathcal{P}_1) \leq^{a_3} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{E}) \leq^{a_3} \partial(\mathcal{P}_3)$ (or equivalenty $\partial(\mathcal{P}_2) \leq^{a_3} \partial(\mathcal{P}_3)$, since $\partial(\mathcal{E}) = \partial(\mathcal{P}_2)$). With (3) and (5), we have

$$\begin{aligned}
\partial(\mathcal{P}_3) &= \left(\frac{a_2+b_2}{b_2}\right)\partial(\mathcal{P}_3) - \frac{a_2}{b_2}\partial(\mathcal{P}_3) \\
&\geq^{a_3} \left(\frac{a_2+b_2}{b_2}\right)\partial(\mathcal{P}_2) - \frac{a_2}{b_2}\partial(\mathcal{P}_1)
\end{aligned}$$

$$= \frac{c_2}{b_2} - \frac{a_2}{b_2}\,\partial(\mathcal{P}_1)$$

$$= \partial(\mathcal{P}_1\mathcal{P}_2) \;>^{a_3}\; \partial(\mathcal{P}_3),$$

a contradiction. We therefore have proved that at least one of $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{E}\mathcal{P}_3$ is an infeasible loop which again contradicts the shortness of $\mathcal{P}$.

II.C) $\mathcal{P}_2$ is permutable and $\mathcal{P}_2 \neq \mathcal{P}_2'$.
Since $\mathcal{P}_2$ is permutable, we have $sgn(a_2b_2)=-1$ and since $\mathcal{P}_2 \neq \mathcal{P}_2'$, there are paths $\mathcal{P}_4$ and $\mathcal{P}_5$ such that $\mathcal{P}_4$ has x as last vertex and $\mathcal{P}_5$ has initial vertex x ($\mathcal{P}_2=\mathcal{P}_4\mathcal{P}_5$ and $\mathcal{P}_2'=\mathcal{P}_5\mathcal{P}_4$). The residues of $\mathcal{P}_2$ and $\mathcal{P}_2'$ are

$$\begin{aligned} r(\mathcal{P}_2) &= r(\mathcal{P}_4\mathcal{P}_5) = \langle a_4, b_4, c_4\rangle * \langle a_5, b_5, c_5\rangle \\ &= sgn(a_5)\,\langle a_4a_5, -b_4b_5, c_4a_5 - c_5b_4\rangle \\ r(\mathcal{P}_2') &= r(\mathcal{P}_5\mathcal{P}_4) = \langle a_5, b_5, c_5\rangle * \langle a_4, b_4, c_4\rangle \\ &= sgn(a_4)\,\langle a_5a_4, -b_5b_4, c_5a_4 - c_4b_5\rangle . \end{aligned}$$

Observe that

$$\begin{aligned} a &= sgn(a_4)\,(a_5a_4 - b_5b_4), \\ c &= sgn(a_4)\,(c_5a_4 - c_4b_5), \\ a_2 &= sgn(a_5)a_4a_5, \\ b_2 &= sgn(a_5)(-b_4b_5) = sgn(b_4)b_4b_5, \end{aligned}$$

and $c_2 = sgn(a_5)\,(c_4a_5 - c_5b_4)$.

We distinguish between the two cases

1) $sgn(a_2+b_2) = sgn(a_2)$,

and 2) $sgn(a_2+b_2) = sgn(b_2)$.

II.C.1) $sgn(a_2+b_2)=sgn(a_2)$.
Since

$$\begin{aligned} & sgn(b_4a) \\ &= sgn(b_4\,(sgn(a_4)\,(a_5a_4 - b_5b_4))) \\ &= sgn(b_4(a_5a_5)a_4(a_4a_5 - b_4b_5)) \\ &= sgn(b_4a_5)*sgn(a_4)*sgn(a_5(a_4a_5 - b_4b_5)) \\ &= (-1)*sgn(a_4)*sgn(a_2) \\ &= -1, \end{aligned}$$

the loop $\mathcal{P}_1\mathcal{P}_4\mathcal{E}$ is admissible by Lemma A.1. The discriminant of the loop $\mathcal{P}_4\mathcal{E}$ can be replaced by the discriminant of the loop $\mathcal{P}_2$:

$$\partial(\mathcal{P}_4\mathcal{E}) = \frac{c_4a - cb_4}{a_4a} = \frac{c_4(a_4a_5-b_4b_5) - (c_5a_4 - c_4b_5)}{a_4(a_4a_5 - b_4b_5)}$$
$$= \frac{a_4(c_4a_5 - c_5b_4)}{a_4(a_4a_5 - b_4b_5)} = \frac{c_2}{a_2+b_2} = \partial(\mathcal{P}_2)$$

Suppose that both $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{P}_1\mathcal{P}_4\mathcal{E}$ are not infeasible. Then, by Lemma A.4, $\partial(\mathcal{P}_1) \leq^{a_3} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{P}_1) \leq^{a_4} \partial(\mathcal{P}_4\mathcal{E}) = \partial(\mathcal{P}_2)$ Recalling $sgn(a_2) = sgn(a_3) = sgn(a_4)$ and (5), we have

$$\partial(\mathcal{P}_1) = \left(\frac{a_2+b_2}{a_2}\right)\partial(\mathcal{P}_1) - \frac{b_2}{a_2}\partial(\mathcal{P}_1)$$
$$\leq^{a_2} \left(\frac{a_2+b_2}{a_2}\right)\partial(\mathcal{P}_2) - \frac{b_2}{a_2}\partial(\mathcal{P}_3)$$
$$= \frac{c_2}{a_2} - \frac{b_2}{a_2}\partial(\mathcal{P}_3)$$
$$= \partial(\mathcal{P}_2\mathcal{P}_3) <^{a_2} \partial(\mathcal{P}_1),$$

a contradiction. Therefore, at least one of the loops $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{P}_1\mathcal{P}_4\mathcal{E}$ is infeasible contradicting the shortness of $\mathcal{P}$.

II.C.2) $sgn(a_2+b_2) = sgn(b_2)$.
Since

$$\begin{aligned} & sgn(aa_5) \\ &= sgn((sgn(a_4)(a_5a_4 - b_5b_4))a_5) \\ &= sgn(a_5(a_4a_5 - b_4b_5))*sgn(a_4) \\ &= sgn(a_2 + b_2)*sgn(a_2) \\ &= sgn(a_2b_2) \\ &= -1, \end{aligned}$$

the loop $\mathcal{E}\mathcal{P}_5\mathcal{P}_3$ is admissible by Lemma A.1. The discriminant of the loop $\mathcal{E}\mathcal{P}_5$ can be replaced by the discriminant of the loop $\mathcal{P}_2$:

$$\partial(\mathcal{E}\mathcal{P}_5) = \frac{ca_5 - c_5a}{-ab_5} = \frac{(c_5a_4 - c_4b_5)a_5 - c_5(a_5a_4 - b_5b_4)}{-(a_5a_4 - b_5b_4)b_5}$$
$$= \frac{b_5(c_4a_5 - c_5b_4)}{b_5(a_4a_5 - b_4b_5)} = \frac{c_2}{a_2+b_2} = \partial(\mathcal{P}_2)$$

Again, suppose that both $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{E}\mathcal{P}_5\mathcal{P}_3$ are not infeasible. We then have by Lemma A.4 that $\partial(\mathcal{P}_1) \leq^{a_3} \partial(\mathcal{P}_3)$ and $\partial(\mathcal{P}_2) = \partial(\mathcal{E}\mathcal{P}_5) \leq^{a_3} \partial(\mathcal{P}_3)$. Recalling (5), we have

$$\partial(\mathcal{P}_3) = \left(\frac{a_2+b_2}{b_2}\right)\partial(\mathcal{P}_3) - \frac{a_2}{b_2}\partial(\mathcal{P}_3)$$

$$\geq^{a_3} \left(\frac{a_2+b_2}{b_2}\right)\partial(\mathcal{P}_2) - \frac{a_2}{b_2}\partial(\mathcal{P}_1)$$

$$= \frac{c_2}{b_2} - \frac{a_2}{b_2}\partial(\mathcal{P}_1)$$

$$= \partial(\mathcal{P}_1\mathcal{P}_2) >^{a_3} \partial(\mathcal{P}_3),$$

a contradiction. Therefore, one of the paths $\mathcal{P}_1\mathcal{P}_3$ and $\mathcal{E}\mathcal{P}_5\mathcal{P}_3$ is an infeasible loop contradicting the shortness of $\mathcal{P}$.

□

The main theorem:

If $\mathcal{G}$ is a closed graph for $\mathcal{S}$, then the following statements are equivalent:

i) $\mathcal{S}$ is satisfiable

ii) $\mathcal{G}$ has no infeasible simple loop

Proof:

If $\mathcal{S}$ is satisfiable, then, by Lemma 2.2, the set of lequalities labeling $\mathcal{G}$ is, as augmentation of $\mathcal{S}$ by its loop residue lequalities, satisfiable. $\mathcal{G}$ thus has no infeasible simple loop.

If, conversely, $\mathcal{G}$ has no infeasible simple loop, then inductively construct a sequence of numbers v_0', v_1', ..., v_t' and a sequence of graphs $\mathcal{G}_0$, $\mathcal{G}_1$, ..., $\mathcal{G}_t$ where v_0, v_1, ..., v_t are the variables in $\mathcal{S}$ in the following way:

Basis:

Let $v_0' = 0$, $\mathcal{G}_0 = \mathcal{G}$.

Induction Step:

For $0<i\ (\leq t)$ let v_i be any value in the interval $[l_i, u_i]$ and obtain $\mathcal{G}_i$ from $\mathcal{G}_{i-1}$ by adding two new edges from v_i to v_0, labeled $v_i \leq v_i'$ and $v_i \geq v_i'$, respectively, where

$$l_i = max\ \{\ \partial\mathcal{P} \mid \mathcal{P} \text{ is an admissible path from } v_0 \text{ to } v_i \text{ in } \mathcal{G}_{i-1} \text{ and } b\mathcal{P} < 0\ \}$$

and $$u_i = min\ \{\ \partial\mathcal{P} \mid \mathcal{P} \text{ is an admissible path from } v_i \text{ to } v_0 \text{ in } \mathcal{G}_{i-1} \text{ and } a\mathcal{P} > 0\ \}\ .$$ [30]

30 : We assume that for empty sets, $max\{\} = -\infty$ and $min\{\} = \infty$. Observe that for a path $\mathcal{P}$ from v_0 to v_i with $b\mathcal{P}<0$ that $v_i \geq \partial\mathcal{P}$, since $a\mathcal{P}=0$, $\partial\mathcal{P}=c\mathcal{P}/b\mathcal{P}$, and $b\mathcal{P}v_i \leq c\mathcal{P}$ by the residue inequality. In a similiar way, for a path $\mathcal{P}$ from v_i to v_0 with $a\mathcal{P}>0$, we have $v_i \leq \partial\mathcal{P}$, since $b\mathcal{P}=0$, $\partial\mathcal{P}=c\mathcal{P}/a\mathcal{P}$, and $a\mathcal{P}v_i \leq c\mathcal{P}$ by the residue inequality.

With (i) of the following claim we ensure that the v_i' and G_i are well defined:

CLAIM

(i) $l_i \le u_i$ for $1 \le i \le r$

(ii) G_i has no infeasible simple loop for $1 \le i \le r$

Proof

By induction on i.

Basis.

For $i=0$, $G_i = G$ and both (i) and (ii) hold.

Induction Step (i):

Assume $l_i > u_i$. Then, by definition of l_i and u_i, admissible paths Q and R from v_0 to v_i and v_i to v_0, respectively, exist in G_{i-1} with $b(Q) < 0$, $a(R) > 0$, and $\partial(Q) > \partial(R)$. Then the loop QR from v_0 to v_0 is admissible by Lemma A.1 and infeasible by Lemma A.4.

G_{i-1} then has by Lemma A.5 an infeasible simple loop, contradicting (ii) of the induction hypothesis.

Induction Step (ii):

Assume on the contrary that G_i has an infeasible simple loop P. Since G_{i-1} has no such loop, and since the two new edges in G_i form no infeasible loop, P (or its reverse) must be of the form QE, where E is one of the two new edges and Q is a path from v_0 to v_i in G_{i-1}.

By admissibility of QE, $sgn(b(Q)=a(E)) = -1$. Thus, $b(Q) < 0$ for $a(E)=1$ (or $b(Q) > 0$ for $a(E)=-1$, and $a(R) > 0$ for R, the reverse of Q), which implies $l_i \ge \partial(Q)$ (or $u_i \le \partial(R) = \partial(Q)$) by the definitions of l_i and u_i, respectively.

If E is labeled with $v_i \le v_i'$ ($v_i \ge v_i'$), then $a(E)=1$ ($a(E)=-1$) and, by Lemma A.4, $\partial(Q) > \partial(E) = v_i'$ ($\partial(Q) < \partial(E) = v_i'$), contradicting $v_i' \ge l_i \ge \partial(Q)$ ($v_i' \le u_i \le \partial(Q)$).

□ CLAIM

Let $ax + by \le c$ be an inequality of S labeling an edge E in G_r. $P = E_1 E E_2$ forms an admissible loop where E_1 is labeled with $x' \le x$ ($x \le x'$) for $a > 0$ ($a < 0$) and E_2 is labeled with $y' \le y$ ($y \le y'$) for $b > 0$ ($b < 0$). So

$$r(E_1) = \langle 0, -sgn(a), -sgn(a)x' \rangle$$

and

$$r(E_2) = \langle -sgn(b), 0, -sgn(b)y' \rangle.$$

The loop residue $r(\mathcal{P})$ computes to

$$
\begin{aligned}
& r(\mathcal{P}) \\
&= r(\mathcal{E}_1)*r(\mathcal{E})*r(\mathcal{E}_2) \\
&= \langle 0,\ -sgn(a),\ -sgn(a)x' \rangle * \langle a,\ b,\ c \rangle * r(\mathcal{E}_2) \\
&= \langle 0,\ -sgn(a)(-sgn(a)b),\ sgn(a)(-sgn(a)x'a - c(-sgn(a))) \rangle * r(\mathcal{E}_2) \\
&= \langle 0,\ b,\ -ax'+c \rangle * \langle -sgn(b), 0, -sgn(b)y' \rangle \\
&= \langle 0,\ 0,\ -sgn(b)((-ax'+c)(-sgn(b)) - (-sgn(b)y')b) \rangle \\
&= \langle 0,\ 0,\ -ax'+c-by' \rangle .
\end{aligned}
$$

Since by the claim above and Lemma A.5, $\mathcal{G}_r$ has no infeasible loop from v_0 to v_0. We thus have $-ax'-by'+c \geq 0$, or $ax'+by' \leq c$. So the v_i satisfy the lequalities of $\mathcal{S}$.

So we have now proved the reverse direction that $\mathcal{S}$ is satisfiable if $\mathcal{G}$ has no infeasible simple loop.

With the forward direction, we have now completed the proof of the main theorem.

□

REFERENCES:

1. ACKERMANN, W. *"Solvable Cases of the Decision Problem"*, North-Holland Pub. Co., Amsterdam, 1954, pp. 98-103.
2. APSVALL, B., SHILOACH, Y. *" A polynomial-time algorithm for solving systems of linear equalities with two variables per inequality"*, Proc. 20th Ann. Symp. on Foundations of Comp. Sc., San Juan, Puerto Rico, 1979, pp. 205-217.
3. BLEDSOE, W.W., *"A new method for proving certain Presburger formulas"*, Advance Papers 4th Int. Joint Conf. on Artif. Intell., Tibilisi, Georgia, U.S.S.R., Sept. 1975, pp. 15-21.
4. COOPER, D.C., *"Theorem proving in arithmetic without multiplication"*, B. Meltzer and D. Michie, Eds., In: Mach. Intell. 7, American Elsevier, New York, 1972, pp. 91-99.
5. DERSHOWITZ, N., *"Termination of Rewriting"*, J. Symb. Comp., Vol. 3, 1987, pp. 69-116.
6. DOWNEY, P., SETHI, R., TARJAN, R. E., *"Variations of the Common Subexpression Problem"*, J. ACM, Vol. 27, No. 4, Oct. 1980, pp. 758-771.
7. GALLIER, J., NARENDRAN, P., PLAISTED, D., RAATZ, S., SNYDER, W., *"Finding canonical rewriting systems equivalent to a finite set of ground equations in polynomial time"*, Lusk and Overbeek, Eds., In: Proceedings of 9th Conference on Automated Deduction, Springer, Argonne, 1988, pp. 182-196.
8. GOMORY, R. E., *"An algorithm for integer solutions to linear programs"*, R. L. Graves and P. Wolfe, Eds., In: Recent Advances in Mathematical Programming, McGraw-Hill, New-York, 1963, pp. 269-302.
9. HARRISON, M., *"Introduction to Formal Language Theory"*, Addison-Wesley Pub. Co., Reading, Mass., 1978.
10. HOARE, C.A.R., *"An axiomatic basis for computer programming"*, Commun. ACM, Vol. 12, 1969, pp. 576-580.
11. HUET, G., *"Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems"*, J. ACM, Vol. 27, No. 4, 1980, pp. 797-821.
12. JOHNSON, D., *"Finding all the elementary circuits of a directed graph"*, SIAM J. Comput., Vol. 4, No. 1, March 1975, pp. 77-84.
13. KOZEN, D., *"Complexity of finitely represented algebras"*, Proc. 9th Annual ACM Symp. on Theory of Computing, Boulder, Colorado, May 1977, pp. 164-177.

14. KREISEL, G., KREVINE, J. L., *"Elements of Mathematical Logic"*, North-Holland Pub. Co., Amsterdam, 1967, pp. 54-57.

15. LOECKX, J., SIEBER, K., *"The Foundations of Program Verification"*, Teubner, Stuttgart, 1984.

16. NELSON, G., OPPEN, D., *"A simplifier based on efficient decision algorithms"*, Proc. Fifth ACM Symp. on Prog. Langs., Tucson, Ariz., Jan. 1978.

17. NELSON, G., OPPEN, D., *"Fast Decision Procedures Based on Congruence Closure"*, J. ACM, Vol. 27, No. 2, Apr. 1980, pp. 356-364.

18. NELSON, G., OPPEN, D., *"Simplification by Cooperating Decision Procedures"*, ACM Transactions on Prg. Langs. a. Sys., Vol. 1, No. 2, Oct. 1979, pp. 245-257.

19. OPPEN, D., *"A $2^{2^{2^N}}$ upper bound on the complexity of Presburger Arithmetic"*, Ph.D. Th., Univ. of Toronto, Toronto, Canada, 1975.

20. PRABHAKER, M., NARSINGH, D., *"On Algorithms for enumerating all circuits of a graph"*, SIAM J. Comput., Vol. 5, No. 1, March 1976, pp. 90-99.

21. PRATT, V. R., *"Two easy theories whose combination is hard"*, Tech. Rep., MIT, Cambridge, Mass., Sept. 1977.

22. PRESBURGER, M., *"Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zählen in Welchem die Addition als einzige Operation hervortritt."*, Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich Warszawa, Warsaw, Poland, 1929, pp. 92-101.

23. READ, R. C., TARJAN, R. E., *"Bounds on backtrack algorithms for listing cycles, paths, and spanning trees"*, ERL Memo M 433, Electronic Research Lab., Univ. of California, Berkeley, Calif., 1973.

24. SHOSTAK, R., *"An algorithm for reasoning about equality"*, Commun. ACM, Vol. 21, No. 7, July 1978, pp. 583-585.

25. SHOSTAK, R., *"A Practical Decision Procedure for Arithmetic with Function Symbols"*, J. ACM, Vol. 26, No. 2, Apr. 1979, pp. 351-360.

26. SHOSTAK, R., *"Deciding Combinations of Theories"*, J. ACM, Vol 31, No. 1, Jan. 1984, pp. 1-12.

27. SHOSTAK, R., *"Deciding Linear Inequalities by Computing Loop Residues"*, J. ACM, Vol. 28, No. 4, Oct. 1981, pp. 769-779.

28. SHOSTAK, R., *"On the SUP-INF Method for Proving Presburger Formulas"*, J. ACM, Vol. 24, No. 4, Oct. 1977, pp. 529-543.

29. SZWARCFITER, J. L., LAUER, P. E., *"Finding the elementary cycles of a directed graph in O(n+m) per cycle"*, Tech. Rep. No. 60, Univ. of Newcastle upon Tyne, Newcastle upon Tyne, England, May 1974.

30. TARJAN, R. E., *"Efficiency of a Good But Not Linear Set Union Algorithm"*, J. ACM, Vol. 22, No. 2, April 1975, pp. 215-225.

31. TARJAN, R. E., *"Enumeration of the elementary circuits of a directed graph"*, SIAM J. Comput., Vol. 2, No. 3, Sept. 1973, pp. 211-216.

32. YASUHARA, A., *"Recursive Function Theory & Logic"*, Academic Press Inc., New York, 1971.