Integration of Rewriting, Narrowing, Compilation, and Heuristics for Equality Reasoning in Resolution-Based Theorem Proving

Axel Präcklein
Fachbereich Informatik, Universität des Saarlandes
Im Stadtwald 15, 6600 Saarbrücken 11

Contents

Contents i

Extended German Abstract v

English Abstract vii

1 Introduction 1
- 1.1 Equality . . . 1
- 1.2 The Central Role of Completion . . . 3

2 Equality Reasoning – An Overview 5
- 2.1 Basic Termini and Definitions . . . 5
- 2.2 Theory Unification . . . 8
- 2.3 Paramodulation . . . 9
- 2.4 Decomposition . . . 10
  - 2.4.1 RUE-Resolution . . . 10
  - 2.4.2 Decomposition-Based E-Unification . . . 12
- 2.5 Rewriting . . . 15
- 2.6 Restrictions for Paramodulation . . . 17
  - 2.6.1 Superposition . . . 17
  - 2.6.2 Reduction . . . 21
  - 2.6.3 Summary . . . 22
- 2.7 Narrowing . . . 22

3 Variations of the Basic Superposition Calculus 25
- 3.1 Inference Rules . . . 25
  - 3.1.1 E-Resolution via Narrowing . . . 26
  - 3.1.2 Completeness Considerations . . . 28
- 3.2 Restriction Strategies . . . 28
  - 3.2.1 The Superposition Strategy . . . 29
  - 3.2.2 A Clause Graph Strategy . . . 29
- 3.3 Reduction . . . 34
- 3.4 Summary . . . 37

4 Heuristic Control **39**
4.1 Narrowing Control . . . 39
4.1.1 Prerequisites for E-Resolution . . . 41
4.1.2 Adaptation of Difference Reduction Heuristics . . . 42
4.1.3 Restrictions for Narrowing . . . 43
4.2 Completion Control . . . 44
4.2.1 Choosing the Restriction Strategy . . . 45
4.2.2 Reduction Orderings . . . 47
4.2.3 Critical Pair Selection . . . 49
4.2.4 Critical Pair Reduction . . . 51
4.3 Summary . . . 52

5 Compilation: Towards an Equality Reasoning Machine **53**
5.1 A Compilation Example . . . 55
5.2 The Transformation into Mnemonic Assembler Code . . . 57
5.2.1 Abstract Code . . . 58
5.2.2 Terms → Lisp . . . 59
5.2.3 Lisp → Single Functions . . . 60
5.2.4 Single Functions → Abstract Code . . . 61
5.3 Evaluation of an Implementation . . . 62
5.4 Unfailing and Conditional Rewriting . . . 64
5.5 Possible Refinements . . . 64
5.5.1 Compiled Unification . . . 64
5.5.2 Indexing Trees . . . 64

6 Aspects of Integration **69**
6.1 Rules for Building Theorem Provers . . . 69
6.2 Control Through System Architecture . . . 72
6.3 Global Supervision . . . 73
6.4 Integration into an Existing Theorem Prover . . . 75
6.5 Usage of Theory Unification . . . 76
6.6 Towards a Human-Oriented Presentation of Equational Proofs . . . 79

7 Conclusion **81**

A Sparc **83**
A.1 Compound Term and Constant . . . 83
A.2 Variable . . . 84
A.3 Term List Cons Cell . . . 85
A.4 Bindings . . . 86
A.4.1 Setting Bindings . . . 86
A.4.2 Checking Bindings . . . 86

B Commented Examples **87**
B.1 Lusk/Overbeek . . . 88
B.1.1 Example 1 . . . 89
B.1.2 Example 2 . . . 89
B.1.3 Example 3 . . . 90
B.1.4 Example 4 . . . 91
B.1.5 Example 5 . . . 93
B.1.6 Example 6 . . . 94
B.2 Pelletier . . . 95
B.2.1 Example 48 . . . 95
B.2.2 Example 49 . . . 96
B.2.3 Example 51 . . . 96
B.2.4 Example 52 . . . 97
B.2.5 Example 53 . . . 97
B.2.6 Example 54 . . . 106
B.2.7 Example 55 . . . 107
B.2.8 Example 56 . . . 108
B.2.9 Example 58 . . . 109
B.2.10 Example 61 . . . 109
B.2.11 Example 63 . . . 110
B.2.12 Example 64 . . . 111
B.2.13 Example 65 . . . 111
B.2.14 Example 73, Four Pigeons – Three Holes . . . 111
B.3 Other Examples . . . 119
B.3.1 Cancellation . . . 119
B.3.2 Two Square Roots . . . 120
B.3.3 Commutator . . . 122
B.3.4 Group Completion . . . 122
B.3.5 Central Groupoid . . . 123
B.3.6 Z22 . . . 124
B.3.7 Z29 . . . 133

Acknowledgement **157**

Bibliography **159**

Index **173**

Zusammenfassung

Mit der vorliegenden Arbeit wird ein Rahmen zur Integration verschiedener Methoden und Werkzeuge im Bereich des Gleichheitsbeweisens vorgestellt. Die Arbeit konzentriert sich auf die praktischen Aspekte dieser Integration: Wie und für welche Beispiele kann man welche Komponenten des Systems verwenden?

Die Gleichheit ist eine der wichtigsten Relationen in der Mathematik und auch bei der Modellierung der uns umgebenden Realität. Deshalb fungierte sie von Anfang an als ein Fallbeispiel für die Methoden der Künstlichen Intelligenz (KI).

Ausgehend von der Standardaxiomatisierung der Gleichheit für die Logik erster Stufe wurden in zahlreichen Veröffentlichungen unterschiedliche Gleichheitskalküle mit dem Ziel der Mechanisierung auf dem Rechner entwickelt. Dabei sind zwei grundverschiedene Varianten zu unterscheiden, nämlich solche, die versuchen, die Unterschiede zwischen zwei Termen zu reduzieren und solche, die mit Ersetzung von Untertermen arbeiten.

Als Basis für diese Arbeit wurden verschiedene Methoden auf ihre allgemeine Anwendbarkeit und auf Möglichkeiten hin untersucht, sie in das MKRP-System, das in unserer Arbeitsgruppe entwickelt wurde, zu integrieren (Kapitel 2):

- Die konzeptionell einfachste Art des Gleichheitsbeweisens ist die Unifikation, die dazu dient zwei oder mehr Objekte durch geeignete Belegung ihrer Variablen gleichzumachen. Hier gibt es Erweiterungen, die bestimmte Theorien wie Assoziativität oder Kommutativität in diesen Prozeß einbeziehen (Abschnitt 2.2).

- Die Basisregel für ersetzungsorientierte Gleichheitskalküle ist die Paramodulation. Die Paramodulationsregel verhält sich in Kalkülen ähnlich wie die Resolutionsregel, beide lassen sich also gleich ansteuern (Abschnitt 2.3).

- Differenzreduzierende Gleichheitsbeweiser arbeiten häufig durch Termdekomposition, dabei sind bestimmte Einschränkungen, die für die Paramodulation möglich sind, nicht übertragbar. Außerdem hat man bei der Integration von Dekomposition und Resolution das Problem, die beiden Regeln verschieden ansteuern zu müssen, weshalb wir uns dann mehr auf die ersetzungsorientierten Verfahren konzentriert haben (Abschnitt 2.4).

- Eine der stärksten Einschränkungen für die Paramodulation stellt eine für das Theorembeweisen taugliche Abart des Knuth-Bendix-Vervollständigungsalgorithmus' dar. Er sollte den Kern eines für Gleichheitsbeweise geeigneten Beweisers darstellen. Der Knuth-Bendix-Algorithmus arbeitete ursprünglich nur auf unbedingten Gleichungen (Abschnitt 2.5).

- Es gibt inzwischen verschiedene Erweiterungen für bedingte Gleichungen, die alle als Restriktionsstrategien der Paramodulationregel angesehen werden können. Von diesen haben wir mehrere in unserem Beweissystem implementiert (Abschnitt 2.6).

- Die „Narrowing"-Technik stellt eine Möglichkeit zur Verfügung mit dem Knuth-Bendix-Verfahren vervollständigte Gleichungssysteme in einem quasi generischen Theorieunifikationsalgorithmus als Theorie zu verwenden (Abschnitt 2.7).

Die Untersuchung der hier aufgeführten Verfahren gab darüber Aufschluß, welche davon im Kontext unseres vorhandenen Beweissystems verwendet werden können (Kapitel 3).

- Es wurde ein Modus für das System entwickelt, der alle bisherigen Fähigkeiten des Systems erhält und mehrere der vorgestellten Verfahren integriert.
- Die bedingten Vervollständigungsverfahren können durch Ansteuern der Paramodulation und Resolution simuliert werden. Wird nur die Paramodulation in diesem Sinne angesteuert, ergibt sich das konservative, praktisch alle Fähigkeiten erhaltende Verfahren. Durch ein Beispiel wird nachgewiesen, daß dieses Verfahren auch in seiner schwächsten Form unvollständig ist. Dies führt nicht notwendigerweise zu seiner Unbrauchbarkeit, da die Unvollständigkeit in der Praxis eine geringe Rolle spielt.
- Durch die heuristische Verwendung von Narrowing kann man den Vorteil der unterschiedsreduzierenden Verfahren, nämlich relativ mächtige Heuristiken, in das im Grunde ersetzungsorientierte Verfahren einbauen.

Ein weiterer Schwerpunkt dieser Integrationsarbeit ist die Untersuchung, wie viele der im Bereich des automatischen Beweisens entwickelten Heuristiken, insbesondere die für die Dekomposition verwendeten, genutzt werden können (Kapitel 4).

- Die Differenzreduktionsheuristiken können problemlos in die Ansteuerung der Narrowingschritte integriert werden. Sie können sogar im Hinblick auf Beurteilung globaler Strukturen erweitert werden.
- Die verschiedenen Superpositionskalküle wurden auf Unterschiede beim Beweis von ausgewählten Beispielproblemen untersucht.
- Eine Schwierigkeit beim Vervollständigungsverfahren ist die Auswahl der Ordnung, die meistens manuell erfolgt. In unserem System verrichtet ein relativ einfacher automatischer Generierungsalgorithmus seinen Dienst.
- Die üblichen Auswahlverfahren für kritische Paare können in das integrierte System übernommen werden. Anhand eines Beispielkatalogs wurde das Verhalten des Systems bei der Variation mehrerer wichtiger Parameter untersucht.

Zur effizienten Realisierung des vorgeschlagenen integrierten Systems wurde ein Kompilationsansatz für Termersetzungsregeln weiterentwickelt, der den Vervollständigungsalgorithmus direkt auf den kompilierten Termersetzungsregeln ablaufen läßt. Leider sind die Ergebnisse des implementierten Prototyps nicht überzeugend, weshalb wir auch keine Erweiterung für allgemeine Klauseln vornahmen. Dieses negative Ergebnis ist zum einen auf die Wahl der konkreten Maschine zurückzuführen, auf der wir unseren Ansatz testeten, zum anderen darauf, daß die zu optimierende Komponente im Verfahren nicht den erwartet hohen Zeitanteil einnimmt (Kapitel 5).

Bei der Integration gab es die Chance, die Verwendbarkeit weiterer Werkzeuge beim automatischen Beweisen zu untersuchen.

- Es wurde die Nützlichkeit verschiedener Theorieunfikationsalgorithmen geprüft (Abschnitt 6.5).
- Es wurde untersucht, inwieweit Gleichheitsbehandlung in ein Verfahren zur strukturierten Darstellung von Beweisen eingebaut werden kann (Abschnitt 6.6).

Die Intergrationsarbeit führte zu einer Anzahl von Regeln und Vorschlägen, deren Beachtung bei der Implementierung automatischer Beweiser sinnvoll ist (Abschnitte 6.1 bis 6.3).

Im Anhang werden exemplarisch Beweisläufe für die Beispiele aufgeführt, die im Text für zahlreiche Statistiken und Analysen verwendet werden. Der Anhang enthält weitere Analysen und Kommentare.

Abstract

We present a framework for the integration of the Knuth-Bendix completion algorithm with narrowing methods, compiled rewrite rules, and a heuristic difference reduction mechanism for paramodulation. The possibility of embedding theory unification algorithms into this framework is outlined.

Results are presented and discussed for several examples of equality reasoning problems in the context of an actual implementation of an automated theorem proving system (the MKRP-system) and a fast C implementation of the completion procedure. The MKRP-system is based on the clause graph resolution procedure.

The thesis shows the indispensibility of the constraining effects of completion and rewriting for equality reasoning in general and quantifies the amount of speed-up caused by various enhancements of the basic method.

The simplicity of the superposition inference rule allows to construct an abstract machine for completion, which is presented together with computation times for a concrete implementation.

Keywords: Equality reasoning, Knuth-Bendix completion algorithm, narrowing, paramodulation, resolution, compilation.

Chapter 1

Introduction

Equality is a very important relation in mathematics. Its mechanization is an interesting field for the employment of artificial intelligence (AI) methods and it has served as a test bed for the development of general techniques in AI from the very beginnings [Dartmouth Conference 1956]. Equality not only occurs in mathematics. The problem of identity for two different formal expressions arises almost anywhere where aspects of reality are treated in a formal way. So, not surprisingly, the concept of equality is encountered in almost every subfield of artificial intelligence.

1.1 Equality

Following Leibniz equality is construed as a higher order property in the sense that two things are equal if they have the same properties, that is, all unary predicates which hold for these objects are equal:

$$\forall P : P(x) = P(y) \Rightarrow x = y$$

This higher order property can be axiomatized in a first order calculus by the following three axioms that define an equivalence relation and two additional axiom schemata that define the equality predicate which is a congruence relation. This is by now a standard treatment of equality in all introductory text books on logic or mathematics like for example E. Mendelson [Men87], H.-D. Ebbinghaus, J. Flum, and W. Thomas [EFT86], or V. Sperschneider [Spe84]. Like all higher order properties the axioms depend on the concretely specified signature, that is, the function and predicate constants f and P.

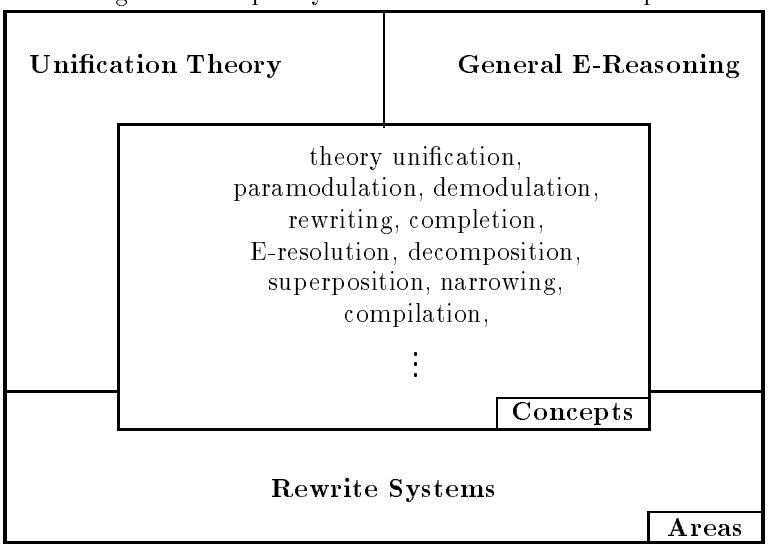
Definition 1.1 (Equality Axioms)

- $\forall x : \ x = x$ *(reflexivity)*
- $\forall x, y : \ x = y \ \Rightarrow \ y = x$ *(symmetry)*
- $\forall x, y, z : \ x = y \wedge y = z \ \Rightarrow \ x = z$ *(transitivity)*
- *For each function symbol f and each argument position i of this function:*
  $\forall x_1, \ldots, x_n, y : \ x_i = y \Rightarrow f(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, y, \ldots, x_n)$
- *For each predicate symbol P and each of its argument positions i:*
  $\forall x_1, \ldots, x_n, y : \ x_i = y \wedge P(x_1, \ldots, x_i, \ldots, x_n) \Rightarrow P(x_1, \ldots, y, \ldots, x_n)$

It would be rather inefficient to handle the equality predicate by these axioms in an automatic theorem prover, nevertheless some trivial problems like B.2.1, page 95 and B.2.2, page 96 could be solved in this way.

As depicted in figure 1.1 research on the mechanization of the equality predicate was historically carried out in several areas, mainly: "General equality reasoning" (see the "International Conferences on Automated Deduction", CADE), "Unification theory" (see the "International Workshops on Unification"), and "Rewriting systems" (see the "International Conferences on Rewriting Techniques and Applications", RTA).

Figure 1.1: Equality Research Areas and Concepts



The first area originated from automated theorem proving and led to a variety of special calculi that are more appropriate for a logical basis of an actual theorem proving system than working with the axioms above. Some of the inference systems, for example paramodulation [RW69] and RUE-resolution [Dig79], integrate a new rule into an existing calculus as for example resolution, others like E-resolution [Mor69] and the more recent general E-unification approaches [Blä86, YS86] ignore the context of the local equality problem and consider only equations without conditions. In fact, they integrate equality completely into the unification algorithm.

Most results in unification theory apply to special equational theories like associativity, associativity with commutativity together, or Boolean rings. Various unification and matching algorithms as well as solvability and complexity issues were dealt with. For an overview see [Sie89].

A new drive for equality reasoning came from the completion method of D. Knuth and P. Bendix [KB70], who constrained the applicability of equations by directing them, such that normal forms of terms are defined. At the beginning, this approach was restricted to orientable unit equations, but by now it has been extended in many directions, for instance to conditional or unorientable equations and to equations modulo a theory.

The main goal of this thesis was to integrate as much as possible of the tools and methods available for equational and resolution theorem proving into one single system. We present a system that is mainly based on three constituents, namely (i) Knuth-Bendix completion extended by clausal superposition, (ii) narrowing which is used to perform difference reduction, and finally (iii) a compilation technique for the application of rewrite rules. All three approaches are integrated into a classical resolution based clause graph theorem prover [Eis88], the "Markgraf Karl Refutation Procedure (MKRP)".

If we examine the combination of these methods, formal descriptions give no evidence for their usability, but only experiments with many examples lead to insights for the feasibility of these methods. So we

are as informal as possible in this thesis and discuss the results of using the algorithms mainly by giving examples and statistics.

In chapter 2 we shall give a general overview of most contemporary equality reasoning methods, and we shall try to motivate our use of these ideas (or why we do not use them). This is followed by a more formal description of the extensions of the inference system of MKRP in chapter 3 that are derived from the equality reasoning methods described in chapter 2. Chapter 4 verifies that the most important heuristics of traditional equality reasoning can be retained within this system. In chapter 5 we concentrate on an efficient implementation, that is, we present an abstract machine for the kernel of the calculus, which is constructed via successive transformations. Chapter 6 outlines the essentials necessary to construct an equational theorem prover from the toolbox and the experiences described in the previous chapters. It illustrates that various methods and tools fit well into the framework of conventional theorem proving.

We see the completion idea and its derivatives as the most important paradigm of equality reasoning and hence as a central part of automated deduction. Based on the examples in the appendix, which cover a wide range of mathematics, we give additional evidence for the importance of this paradigm. Our work is experimental and empirical in nature as opposed to purely theoretical work and hence there is a strong emphasis on actual examples, that are collected in the extensive appendix.

1.2 The Central Role of Completion

Systems which are not based on normal forms for terms, as for example the decomposition approaches of V. Digricoli [Dig79] and K. Bläsius [Blä86], as well as J. Gallier's first system [GS86] often produce many unifiers for the same subproblem. For example they may generate the unifiers $\{x \mapsto 0\}, \{x \mapsto -0\}, \{x \mapsto --0\}$ for the problem $x = 0$, that is, these systems try to enumerate all unifiers $\{\{x \mapsto (-)^n 0\} \mid n \in \mathbb{N}\}$. One possibility that avoids this effect is to postpone the computation until a solution of the subproblem $x = 0$ is really needed. But in computational terms there is a better possibility to circumvent this situation: demodulation can be used to reduce all solutions to a simplest one. The principle of demodulation was introduced and promoted by L. Wos [WRCS67, WOLB84]. Equations are directed and applied to all terms. The orientation can be chosen either heuristically by the machine or else by the user. When we use demodulation it is clear that good demodulators should be computed during the search for a proof. This directly leads to the usage of the Knuth-Bendix completion algorithm with its strong and elaborated reduction orderings.

The key point is that the constraints introduced by the orientation of equations and the reductions are so strong that it is better to perform forward search as opposed to goal oriented backward search, which is performed in a decomposition approach as we shall see. This forward reasoning style was one of the main reasons for the lack of confidence of classical automated deduction research into this method.

The following sketch of an example gives some intuitive evidence for the drastic reduction of the search space in the completion method. Many authors as for example J. Siekmann [Sie75], A. Bundy [Bun83], and K. Bläsius [Blä86] discussed the complexity of automatically finding a proof for the problem "every group with $x + x = 0$ is commutative" (see example B.1.1). Their comparison of the different calculi is based on the number of inference steps that are necessary to solve the problem.

A resolution theorem prover with explicit usage of the equality axioms as stated in definition 1.1 based on breadth first search must generate about 10^{21} resolvents to prove this theorem. For a similarly uninformed paramodulation prover the situation is "slightly" better, but it still has to create approximately 12^{10} ($\approx 6 \cdot 10^{10}$) clauses. This reduction in the number of steps stems from the fact that paramodulation search trees are not as deep as resolution search trees. But they are much more bushy, and for that reason the amount of reduction is far less than originally hoped for by the inventors of the paramodulation rule.

It is intuitively clear that an orientation of the equations (that is, their usage in only one direction and their usage as normalization rules) leads to another reduction. However, this reduction is far more drastic than usually anticipated: if we regard each selection of a critical pair and its conversion into a rule as one step, we arrive at the following computation: Administrating the critical pairs with a FIFO-strategy, which simulates the breadth first search, leads to a proof in less than 100 steps. With some simple heuristic refinements the number of steps is reduced to 7 (Example B.1.1).

This example alone may show that completion is indispensable for equality reasoning and that it should be placed into the centre of any efficient equality reasoning program. The question arises why this approach was ignored in many theorem proving systems based on traditional methods of automated reasoning.

One standard objection against the application of rewriting systems pointed to the fact that a completion approach is at face value limited to a few special cases, where a canonical rewriting system can be obtained. But even if the Knuth-Bendix procedure diverges, enough interesting intermediate results can be derived as exemplified in problems B.1.3 and B.1.5. These examples demonstrate that most of the generated equations are often useful as *lemmata* to find the proof for the theorem. In such cases completion is superior to any other method dealing with equality.

Another disadvantage of completion as mentioned above appears to be its forward reasoning style without a goal. Furthermore there is no way to distinguish between different abstraction levels: the only parameters that can be set are the reduction ordering and the selection strategy for choosing critical pairs to be directed. But this defect is more than compensated for by the constraining effect of the orientation. For problems which combine equality and other predicates forward reasoning in such a way saves resources, because the computation of the lemmata is only done once and they can then be used a hundred or a thousand times in the remaining search process.

Chapter 2

Equality Reasoning – An Overview

We begin this chapter with the basic definitions for terms, clauses, and other symbolic objects that are standard in the field. Then we give an overview of the context of our work. This seems appropriate in view of the more practical approach we take in this thesis: the main goal of our work is to integrate these methods into one single system and to evaluate the different approaches.

We think that the historical division of equality reasoning into general equality reasoning, unification theory, and term rewriting is inappropriate for structuring the whole research area. This is especially so, because there are many interdependencies and parallel developments in these areas. For some concepts the assignment to the area is not at all straightforward: narrowing, for example, is a technique that belongs conceptually to unification as well as to rewriting.

Another possibility to classify equality reasoning was given by K. Bläsius [Blä86]: two main principles can be distinguished, namely *superposition* (*subterm replacement*) and *difference reduction* (see also section 6.3). Paramodulation, for example, works according to the first principle, where a subterm of a term is replaced by some other term – maybe with some restriction strategy but almost always without any general plan. In contrast difference reduction tries to minimize and finally remove the difference between two terms using special operations similar to those employed in GPS [NSS59].

These two different techniques lead to two different views of resolution and paramodulation. The one related to superposition considers resolution as a subcase of paramodulation. The other view completely eliminates the paramodulation rule in the extremal case and extends unification and resolution to E-resolution. Unfortunately, this classification does not clarify any of the dependencies of the available techniques either.

Hence we choose to present the techniques and concepts actually available to build programs which are able to adequately handle the equality predicate. In some cases the name of a method is the same as that of a research area as can be seen in figure 1.1.

In the following sections 2.1 and 2.2 we shall give some basic definitions. The sections 2.3 and 2.4 introduce the rules for replacement and decomposition based equality reasoning. In the rest of the chapter we elaborate on some useful extensions and refinements of the replacement (rewriting) approach.

2.1 Basic Termini and Definitions

Definition 2.1 (Variable, Function, Term)

$\mathcal{F} = \bigcup \mathcal{F}_n$ *is a denumerable set of* **function symbol**s*, called a signature and consisting of disjoint subsets of function symbols with arity* $n \geq 0$*. Function symbols with arity 0 are called* **constant symbol**s*.*

$\mathcal{V}$ is a denumerable set of **variable symbol**s.

$\mathcal{T} = \mathcal{T}(\mathcal{V},\mathcal{F})$ is the set of **term**s, *that is, the least set with $\mathcal{V} \subseteq \mathcal{T}$, and $f_n t_1 \ldots t_n \in \mathcal{T}$, whenever $f_n \in \mathcal{F}_n$ and all $t_i \in \mathcal{T}$.*

In the sequel we sometimes write the terms with parentheses if they occur within the written text, but in examples and figures the parentheses often hide more information than they transmit.

Definition 2.2 (Predicate, Atom, Formula)

$\mathcal{P} = \bigcup \mathcal{P}_n$ is a denumerable set of **predicate symbol**s, *consisting of disjoint subsets of predicate symbols with arity $n \geq 0$.*

$\mathcal{A} = \{P_n(t_1,\ldots,t_n) | n \geq 0, P_n \in \mathcal{P}_n$ and all $t_i \in \mathcal{T}\}$ is the set of **atom**s.

Φ is the set of **formula**e, *that is, the least set with $\mathcal{A} \subseteq \Phi$, and $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$, $(A \Leftrightarrow B)$, $(\neg A)$, $(\forall x A)$, $(\exists x A) \in \Phi$ whenever A and $B \in \Phi$.*

We often omit the parentheses in atoms and formulae for the sake of clarity, as mentioned above.

Definition 2.3 (Literal, Clause)

If $A \in \mathcal{A}$ then A and $\neg A$ are **literal**s.

Special flattened formulae $L_1 \vee L_2 \vee \ldots \vee L_n$ with all L_i literals are **clause**s. *They are often written as set $\{L_1, L_2, \ldots, L_n\}$. We also use the recursive notation $L \vee C$ for clauses constructed from a literal L and a clause C.*

Definition 2.4 (Substitution, Unifier, Matcher)

A **substitution** *is an endomorphism $\sigma : \mathcal{T} \rightarrow \mathcal{T}$, such that the set $\{x \in \mathcal{V} | \sigma(x) \neq x\}$ is finite. The empty substitution is denoted by ε, all others by their finite set of variable-term pairs $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$. $\{x_1,\ldots,x_n\}$ is called the* **domain**, *$\{t_1,\ldots,t_n\}$ the* **codomain** *of the substitution.*

The composition of two substitutions is written $\sigma\tau$. $\sigma = \tau$ iff $\sigma(t) = \tau(t)$ for all $t \in \mathcal{T}$. $\sigma \geq \tau$ iff there exists a substitution ρ with $\sigma = \rho\tau$.

Given a set of terms $\{s_1,\ldots,s_m\}$ we call a substitution σ with $\sigma(s_1) = \cdots = \sigma(s_m)$ a **unifier** *of $s_1,\ldots,s_n$.*

σ is a **most general unifier** *(mgu) of $\{s_1,\ldots,s_m\}$ if for all unifiers τ of $\{s_1,\ldots,s_m\}$ $\tau \geq \sigma$.*

Given two terms t and s we call a substitution σ with $\sigma(t) = s$ a **matcher**, *which matches t on s.*

Definition 2.5 (Position, Subterm, Replacement)

The **position**s *of a term t are denoted by $\Pi(t)$ and are defined as follows:*

$$\Pi(t) = \begin{cases} \{\varepsilon\} & \text{if } t \in \mathcal{V} \\ \{ip | i \in \mathbb{N}, 1 \leq i \leq n, p \in \Pi(t_i)\} \cup \{\varepsilon\} & \text{if } t = f(t_1,\ldots,t_n), n \in \mathbb{N}, n \geq 0 \end{cases}$$

To every position $p \in \Pi(t)$ corresponds a unique **subterm** *of t denoted by $t|p$.*

We extend the notion of positions canonically to literals and formulae.

$t[p \leftarrow s]$ denotes the **replacement** *of the subterm $t|p$ of t by s.*

Example 2.6 (Position, Subterm, Replacement)

$\Pi(f(a,g(x))) = \{\varepsilon, 1, 2, 21\}$

$\Pi(a) = \{\varepsilon\}$

$f(a,g(x))[21 \leftarrow g(b)] = f(a,g(g(b)))$

$\neg P(a)[11 \leftarrow b] = \neg P(b)$

In accordance with N. Eisinger [Eis88] we define deduction and reduction rules. M. Bonacina and J. Hsiang [BH91] named them expansion and contraction rules, respectively. For the actual definitions of rules we use the following schemata 2.7 and 2.8.

Definition 2.7 (Deduction Rule Scheme)

$$\begin{array}{l} Clause_1 \\ \vdots \\ Clause_n \quad \textit{if} \quad Condition \\ +\underline{\qquad\qquad} \\ Clause_{n+1} \end{array}$$

This scheme means that $Clause_{n+1}$ can be derived from the clauses $Clause_1$, ..., $Clause_n$. The + indicates that Clause is added to the existing set of clauses. Condition specifies when the rule can be applied.

Definition 2.8 (Reduction Rule Scheme)

$$\begin{array}{l} Clause_1 \\ \vdots \\ Clause_n \quad \textit{if} \quad Condition \\ Clause \\ \downarrow\underline{\qquad\qquad} \\ Clause' \end{array}$$

This scheme means that Clause′ can be derived from Clause using $Clause_1 \ldots Clause_n$. The ↓ indicates that Clause is replaced by Clause′ rather than added, so that the total number of clauses remains the same.

Thus we can define the usual binary resolution rule as follows:

Definition 2.9 (Resolution Step)

$$\begin{array}{l} L_0,\ L_1,\ \ldots,\ L_n \\ \neg K_0,\ K_1,\ \ldots,\ K_m \\ +\underline{\qquad\qquad\qquad\qquad} \quad \textit{if} \quad \sigma \textit{ is an mgu of } L_0 \textit{ and } K_0. \\ \sigma(L_1,\ \ldots,\ L_n,\ K_1,\ \ldots,\ K_m) \end{array}$$

Now we have a look at the particulars of the clause graph calculus, which is inter alia the basis for the Markgraf-Karl system [BBB+84, OS89, EOP89, Eis88]. A clause graph consists of a set of clauses, each of them a multiset of literals, and a set of links, which connect pairs of literals with unifiable atoms. A link connecting a positive and a negative literal is called an **R-link** (resolution), while an **S-link** (subsumption) joins two literals with the same sign. If the literals incident with a link belong to two different clauses, it is an **R2-link** or **S2-link**. If both literals belong to the same clause, the link is called an **R1-link** or an **S1-link**. If the atoms of two literals are unifiable only after renaming their variables apart we speak of a weak link.

The different kinds of links provide immediate access to different kinds of operations involving a given literal occurrence. Most notably, R2-links represent the possible applications of the resolution rule, and S1-links indicate factoring. Sets of compatible S2-links represent subsumption possibilities. When applying such deduction rules, we have to add to the graph the new clause along with the links connecting the new literals to the existing graph.

If the new literals are instances of ancestor literals already present in the graph, the new links can be obtained without search by a simple inheritance process. This inheritance was invented by R. Kowalski [Kow75] and extended to R1-links by M. Bruynooghe [Bru75]. For a detailed explanation of the mechanisms in the MKRP-system see H. J. Ohlbach [Ohl87] or H. J. Ohlbach and J. Siekmann [OS89]. The

transfer to S-links is trivial. To remove tautologies or subsumed clauses, special link conditions must be fulfilled.

For new literals that are not obtained by instantiating others, for example the paramodulated literal in a paramodulation step, this form of link inheritance unfortunately does not work [Blä86] (see also section 3.2.2).

In the case of paramodulation there are also approaches based on links and inheritance. In J. Siekmann's and G. Wrightson's paper [SW80], for example, links to be paramodulated upon do not join literals, they join one side of a positive equation with an arbitrary unifiable term in another literal. They are P2-links if the other literal is in a different clause, P1-links if they are in the same. An inheritance mechanism for such links was implemented in our system, but unfortunately P-link inheritance does not work as easily as for R-links because after each resolution or paramodulation step unifiers are applied and therefore completely new terms are generated (see theorem 3.7). Hence our first task was to repair this inheritance mechanism to produce the lacking links. This is simply done by newly generating all P-links and all problematic other links, but of course this brute force approach violates the essential motivation for the clause graph procedure.

2.2 Theory Unification

The simplest case of working with equality is to make two objects equal by the consistent replacement of subobjects by others. Unification is the process of finding a uniform replacement for the variables such that the terms to be unified become syntactically equal, which means that they can be written as the same string. The endomorphism describing the replacement for the variables is called a substitution. As defined in section 2.1 a unifier is a substitution which makes the terms equal. For example $\{x \leftarrow b, y \leftarrow a\}$ is a unifier of $f(x,a)$ and $f(b,y)$, but $f(a,x)$ and $f(b,y)$ are not unifiable.

The classification of equality reasoning methods into replacement and decomposition based approaches originated from unification theory: decomposition was first used by J. Herbrand in his thesis [Her30], whereas J. Robinson proposed a replacement based unification algorithm for his resolution principle [Rob65].

To come closer to the mathematical equality relation the notion of unification can be extended to **E-unification**, where a set E of equations is given as axioms, which induce an equivalence relation that is written $=_E$. An example for a unifier of $f(a,x)$ and $f(b,y)$ under the theory $E = \{f(x,y) = f(y,x)\}$ of commutativity is $\{x \leftarrow b, y \leftarrow a\}$.

In general a unification problem can have more than one solution. J. Robinson [Rob65] proved that in the case of the empty theory E there exists (up to the renaming of variables) a unique **most general unifier** (mgu) representing the whole set of solutions whenever this set is not empty. In arbitrary theories there is not necessarily such a unique representative unifier. The next step was to extend the concept to sets of unifiers, which fulfill the requirements to be correct, complete, and minimal. But there are theories for which such sets do not exist either. Hence equational theories can be classified as to whether for each unifiable set of terms the set of most general unifiers has only one element, is finite, infinite, or does not exist at all, that is, it cannot be distinguished from the set of all unifiers [Sie89]. The corresponding theories are called unitary, finitary, infinitary, and nullary.

One task in unification theory is to develop algorithms to compute sets of unifiers. A universal unification algorithm is an algorithm that works for all theories [Sie89], usually this notion is also used for algorithms handling whole classes of theories (for example [SS81]). Another main goal in this area is to combine known unification algorithms for special theories to obtain new ones for more complex theories. However there are problems: for example the algorithms for associative unification and commutative unification could not be combined to an algorithm for theories that have both properties, and this is the case for almost all theories. In general a new algorithm must be designed for such combined theories. M. Schmidt-Schauß' method for the combination of unification algorithms [Sch89] works for arbitrary

disjoint theories and free function symbols. The newest publication on this topic is an extension which can additionally serve to combine decision procedures, that is, unification algorithms for some infinitary theories. It has been developed by F. Baader and K. Schulz [BS92].

In some cases theory unification algorithms are powerful tools to prove theorems which cannot be proved in any other way, but there are examples where they slow down the search for a proof, because the unification process itself becomes very time consuming. These points are discussed in section 6.5 (see also A. Mahn [Mah91]).

C. Kirchner's approach [Kir85] induces the idea to automatically or manually construct new unification algorithms. This is especially worthwhile for frequently occurring theories but we did not integrate any of these into our system. It appears that theory unification is not compatible with the compilation approach as presented in chapter 5, at least not with our current understanding.

2.3 Paramodulation

A general purpose deduction system must handle all combinations of equations, even if they occur together with other predicates in the same formula, as for example in $\forall n : Even(n) \Leftrightarrow (\exists m : n = 2m)$.

The handling of equality via the axioms in definition 1.1 is very inefficient and this motivated inter alia J. Darlington [Dar68], E. Siebert [Sib69], J. Robinson [Rob65], and G. Robinson and L. Wos [RW69] to incorporate the equality relation into automated deduction systems by designing new inference rules. The best known such inference rule is paramodulation, which works on two clauses one of which contains a positive equality literal. One side of the equation must be unifiable with a subterm t in the other clause by a substitution σ. Then the paramodulant consists of all literals of the two clauses without the equality literal after replacing the term t by the other side of the equation and applying the substitution σ to all literals of the new clause. R. Kowalski showed [Kow75] the advantages of the paramodulation rule and how it can enhance the power of a deduction system.

Definition 2.10 (Paramodulation)

$$\frac{\begin{array}{l} L_0,\ L_1,\ \ldots,\ L_n \\ t = s,\ K_1,\ \ldots,\ K_m \\ + \end{array}}{\sigma(L_0[p \leftarrow s],\ L_1,\ \ldots,\ L_n,\ K_1,\ \ldots,\ K_m)} \quad \textit{if} \quad p \textit{ is a position in } L_0 \textit{ and } \sigma \textit{ is an mgu of } L_0|p \textit{ and } t^{(ii)}.$$

If equality is embedded into resolution theorem proving via the paramodulation rule the clause $x = x$ must be added. However, the application of paramodulation can be restricted in such a way that it is never necessary to paramodulate from and into variables, one of the main factors for the immense search space. G. Robinson and L. Wos showed that this inference system is sound and complete in combination with the resolution rule and the functional reflexive axioms [RW69][(iii)]. The functional reflexive axioms were shown to be superfluous by D. Brand [Bra75], for this fact see also M. Richter [Ric78]. Unfortunately these restrictions cause the incompleteness of the set-of-support strategy[(iv)], and therefore it is difficult to construct a goal oriented equational calculus[(v)].

Paramodulation is a deduction rule that is applicable "almost everywhere" making search graphs very bushy [Bun83] (see section 1.2), and so it should only be used if the result is of overriding importance for other arguments in the proof. One such application of paramodulation could be to transform two literals into resolvable ones. That was the motivation for the so-called E-resolution principle [Mor69, And70].

(ii) Of course $=$ is considered to be symmetric and hence it does not matter whether to paramodulate with s or t.

(iii) $f(x) = f(x)$, for example, is the functional reflexive axiom for the function symbol f.

(iv) The unsatisfiable clause set $\{f(a,b) = a, a = b, f(x,x) \neq x\}$ with $\{f(x,x) \neq x\}$ as set-of-support can only be refuted by paramodulating into a variable [SL91b].

(v) There are also a lot of non equality examples where the set-of-support strategy is not at all the best choice (see page 50).

Definition 2.11 (E-Resolution)

Let $A, B, C, D_1, \ldots, D_n$ be clauses. Then the following two rules deduce E-resolvents, where $P(s_1, \ldots, s_m)$ and $\neg P(t_1, \ldots, t_m)$ are the literals to be resolved upon, and $l_i = r_i$ for $1 \leq i \leq n$ are the defining equations of the theory.

$$\begin{array}{l} P(s_1, \ldots, s_m) \vee A \\ \neg P(t_1, \ldots, t_m) \vee B \\ l_1 = r_1 \vee D_1 \\ \vdots \\ l_n = r_n \vee D_n \\ + \hline \sigma(A \vee B \vee D_1 \vee \ldots \vee D_n) \end{array} \quad \textit{if} \quad s_1 = t_1, \ldots, s_m = t_m \text{ can be shown to be valid using } \sigma l_1 = \sigma r_1, \ldots, \sigma l_n = \sigma r_n.$$

The second rule is:

$$\begin{array}{l} l \neq r \vee C \\ l_1 = r_1 \vee D_1 \\ \vdots \\ l_n = r_n \vee D_n \\ + \hline \sigma(C \vee D_1 \vee \ldots \vee D_n) \end{array} \quad \textit{if} \quad l = r \text{ can be shown to be valid using } \sigma l_1 = \sigma r_1, \ldots, \sigma l_n = \sigma r_n$$

.

This can be seen as a special case of the theory resolution rule, defined by M. Stickel [Sti85], with an undecidable theory. The question of how to prove the validity of the equations $s_1 = t_1, \ldots, s_m = t_m$ is left open in the formulation of theory resolution. J. Morris [Mor69] uses a variant of paramodulation where the intermediate results are stored in trees. So E-resolution is an approach to use paramodulation more goal oriented, but in the conclusion of section 2.4 we shall argue that the resolution steps are not the mile stones of equational proofs anyway: they are at best something like second-rate steps, because they typically correspond to comparatively trivial conclusions. Therefore there are many practical reasons to use a calculus with paramodulation as the dominant rule and resolution only as a secondary rule.

As we shall see paramodulation can be the frame to incorporate several general equality reasoning strategies into one system. We shall elaborate on this concept in chapter 3.

2.4 Decomposition

Having introduced the basic rules for replacement based equality reasoning in the previous section we now turn to difference reduction techniques, the second type of equality reasoning according to K. Bläsius. A decomposition approach to automated theorem proving, which is based on difference reduction, was first advocated and elaborated by V. Digricoli [Dig79].

2.4.1 RUE-Resolution

V. Digricoli extended the resolution calculus by defining the two additional inference rules RUE (resolution by unification and equality) and NRF (negative reflexive function) on the same abstraction level as the resolution rule. He used disagreement sets to define the inference rules.

Definition 2.12 (Disagreement Sets)

A **disagreement set** *for two terms s and t is defined recursively:*

If s and t are equal, the unique disagreement set is $\emptyset$.

If s and t differ at top level, the unique disagreement set is $\{[s,t]\}$.

If $s = fs_1 \ldots s_n$ and $t = ft_1 \ldots t_n$ then $\{[s_i,t_i] | s_i$ and t_i are different$\}$ is a disagreement set and all $\bigcup_{1 \leq i \leq n} D_i$ are disagreement sets, with each D_i a disagreement set of s_i and t_i[(vi)]*.*

Definition 2.13 (RUE)

Let A and B be clauses:

$$\frac{P(s_1,\ldots,s_m) \vee A \quad \neg P(t_1,\ldots,t_m) \vee B}{+\quad \sigma(A \vee B \vee D_{\mathrm{RUE}})} \quad \textit{if}$$

D_{RUE} is the disjunction of inequalities $s \neq t$ with $[s,t]$ ranging over the elements of a selected disagreement set of $\sigma(P(s_1,\ldots,s_m))$ and $\sigma(P(t_1,\ldots,t_m))$ with an arbitrary substitution σ[(viii)]*.*

Definition 2.14 (NRF)

Let C be a clause:

$$\frac{l \neq r \vee C}{+\quad \sigma(C \vee D_{\mathrm{NRF}})} \quad \textit{if}$$

D_{NRF} is the disjunction of inequalities $s \neq t$ with all $[s,t]$ in one disagreement set of $\sigma(l)$ and $\sigma(r)$ with an arbitrary substitution σ[(viii)]*.*

These rules perform a difference reduction strategy that embeds equality unification into the deduction and not into the unification method. As these definitions show, part of the unification process, namely the successive computation of the disagreement sets, is performed on the same level as resolution. RUE and NRF steps really work on terms and not on literals and hence are "smaller" than E-resolution steps. This solves the control problem of E-resolution mentioned above. It has the advantage that the steps can be performed more often but it has the disadvantage that many new clauses can be brought in and the deduction procedure, which controls the application of the resolution rule, is overstrained with decisions concerning the equality predicate. In this way the concept of theory resolution is diluted. No orientation of equations is provided.

Hence, in contrast to J. Morris, V. Digricoli uses no separate new unification method but puts the equality inference rules on a par with the resolution rule.

V. Digricoli [Dig85] gives a set of heuristics to control the application of RUE and NRF. He uses syntactic heuristic rules, which are generally applicable to all equality problems or to resolution like inference rules. These heuristic rules are also used and extended by K. Bläsius (see the next section).

(vi) Hence there can be several disagreement sets for two terms.

(viii) V. Digricoli proposes the usage of special substitutions σ (*most general partial unifiers*) but the completeness proof is based on arbitrary ones and can only be restricted to using empty substitutions. Hence the selection of the unifier has to be driven heuristically.

2.4.2 Decomposition-Based E-Unification

The concept of decomposition for E-unification is a method to derive unifiers for the subterms of the given terms, and to combine these solutions to solve the equality problem for the whole term.

A. Martelli and U. Montanari [MM82] exploited the "divide and conquer" strategy of J. Herbrand for an alternative to the unification algorithm of J. Robinson [Rob65]. The kernel of the unification algorithm based on decomposition is given in definition 2.15.

Definition 2.15 (Unification by Transformation Rules)
A unification problem is a set of equations. It is in solved form when each equation has the form $x = t$ with the variable x not occurring anywhere else in the equation set. The following rules are performed on a set of equations until no rule is applicable. If the system is in solved form this is the final situation and the derived set of equations represents a solution, else no solution exists.

1. *Switching: replace an equation $t = x$ by $x = t$.*
2. *Deletion: delete $t = t$.*
3. *Decomposition: replace $fs_1 \ldots s_n = ft_1 \ldots t_n$ by $s_1 = t_1, \ldots, s_n = t_n$.*
4. *Elimination: if $x = t$ is an equation where x does not occur in t, then replace the occurrences of x by t in all other equations.*

A. Martelli and U. Montanari refined this version using special data structures and labelings and obtained an almost linear unification algorithm. Of course they used a different representation of unifiers because the exponentiality of Robinson-unification stems from the term replacement property of idempotent unifiers, that is, the algorithm to construct the unifiers is itself exponential.

One advantage of the usage of nondeterministic rules is that the order of operations is easier to control and unimportant conditions need not be checked in the control mechanism. This can make soundness and completeness proofs for theory unification algorithms much easier.

C. Kirchner [Kir85] invented a conceptual framework to include special equality theories in such a rule based algorithm. J. Gallier and W. Snyder [GS89] and K. Bläsius [Blä86] concurrently described universal unification algorithms via rules. J. Gallier used a Martelli-Montanari-like version for the pure unification part, whereas K. Bläsius unifies with a Robinson procedure. In addition K. Bläsius' approach is more implementation oriented and proposes special graph structures for storing the information about the unification state. Both do not handle equations with conditions and both have a substantial disadvantage against superposition oriented equational reasoning: they need functional reflexive axioms. In J. Gallier's and W. Snyder's system a restricted form of those axioms is put in the rule "root imitation" and is constrained to be applied only to the top level of terms. It can be regarded as variable abstraction of all subterms of a compound term.

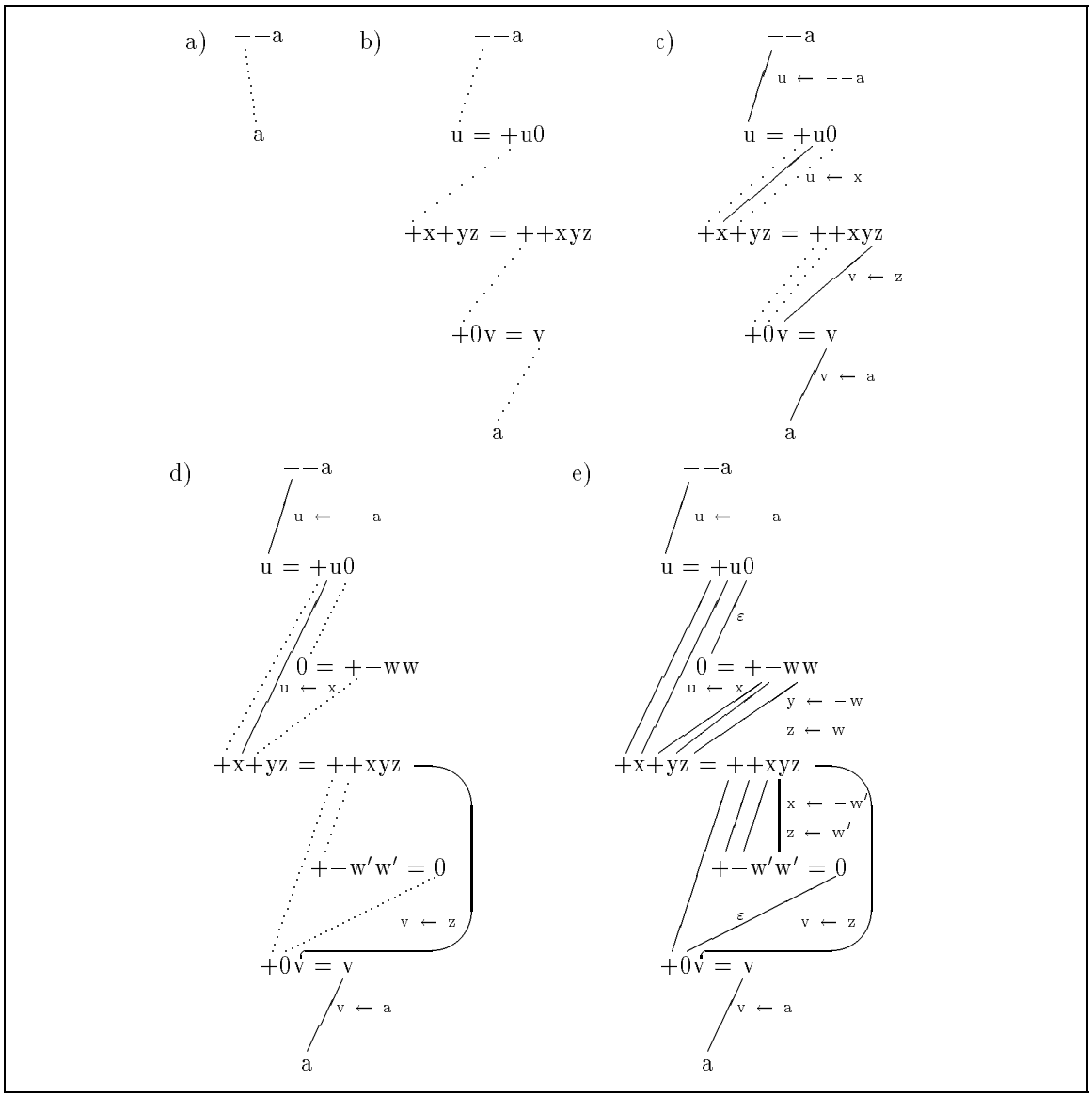
In their improved system they relax the constraint of top level application of equations, and as a result they do no longer need the functional reflexive axioms. But even in this system they cannot avoid to use rewrite rules in the opposite direction, the introduced reduction ordering can only be used when the set of equations and rules is known to be ground Church-Rosser. For the improved system they use an extension of narrowing (see section 2.7).

D. Dougherty and P. Johann [DJ90] constrained the Gallier-approach to the usage of narrowing instead of paramodulation when applying equations.

We implemented an equality reasoning system using an improved version of K. Bläsius' rules. First the Robinson unification rules were replaced by Martelli-Montanari-rules, because these fit better into the general framework due to their decomposition nature.

We demonstrate the usage of the rules with the help of a classical example, namely that $--x = x$ in a group. Unsolved subproblems are indicated by dashed lines, solved subproblems by complete lines labeled with unifiers. Equality chains are represented as $term_1 \text{ — } l_1 = r_1 \text{ — } \cdots \text{ — } l_n = r_n \text{ — } term_2$, two terms concatenated with — must always have the same top function symbol or must be variables. No equation is allowed to occur more than once in one chain. Otherwise it is possible to concatenate multiple chains.

Figure 2.1: Example, Difference Reduction



Example 2.16 (Group, Involution) *The figures* 2.1a *through* 2.1e *depict the development of a graph for the equation* −−a = a *in a group.*

In figure 2.1a *the initial termgraph is shown, it represents that* −−a *and* a *are to be made equal. The dashed line indicates the problem to be solved.*

Figure 2.1b *shows the graph after the insertion of the equality chain* u = +u0 — +x+yz = ++xyz — +0v = v. *Four subproblems indicated by the dashed lines must be solved and their solutions must be*

combined to solve the whole problem. The chains to be inserted must have the property that the terms of each subproblem have the same top level symbol. In this case the pairs of toplevel symbols of the terms are u — −, + — +, + — +, *and* v — a.

Figure 2.1c *shows the graph after the solution of the first and fourth subproblems and the decomposition of the second and third. Two of the new subproblems can be solved trivially. Note that the subproblems* 0 = +yz *and* +xy = 0 *are structurally equal.*

In figure 2.1d *the solved subproblems are indicated by lines marked with the corresponding unifier. Two chains can be inserted to solve the nontrivial subproblems of the last graph.*

In the final graph, depicted in figure 2.1e, *all subproblems were solved and the unifiers can be successively combined to derive a substitution that is the empty substitution* ε *if restricted to the variables of the input terms.*

In this example only the successful steps of the algorithm are depicted. However, as always, there is an enormous amount of useless steps in the search space. The power of an equality prover lies in its ability to avoid such useless steps as often as possible. Even the duplicate steps, like for example the second one with the axiom +−ww= 0 in the above example, should be avoided.

K. Bläsius and V. Lotz [Lotz88] used several heuristics in the implementation of their system and the main power of the program stems from these heuristics. But the results are unsatisfiable when compared to the standard problems of equality reasoning: only theorems of the difficulty like the rather simple examples B.1.1 and B.1.2 proposed by E. Lusk and R. Overbeek [LO84] could be solved. One main reason is that operations with variables are necessary as can be seen in figure 2.1.

Refinements like the following one resemble "dynamic programming" in software development and "indexing" in theorem proving, but they enhance the power of the method only slightly. To use different solutions of structurally identical subproblems at different positions in the graph all subproblems with their graphs can be organized in a hashtable. The hashkey is computed from the structure of the two terms of the equality problem. The test for equality of two such pairs of terms (two equality problems) is made efficient by using the same variable, theory-free constant, and theory-free function symbols, that is, $fc_1c_2 = fc_2x$ is structurally the same problem as $fc_2c_1 = fc_1y$ when c_1 and c_2 are Skolem constants (not occurring in a theory) and x and y are variables. This approach is similar to the usual indexing mechanisms in automated theorem proving [OL80, Ohl89]. In the context of compilation techniques such stored result objects are often called "suspensions" [CL91]. Everywhere in the graph a "renaming" to the standard representation is stored instead of commonly used subproblems, in the example just given we have: $\{x \leftarrow x_1\}$ and $\{c_2 \leftarrow c_1, c_1 \leftarrow c_2, y \leftarrow x_1\}$. Solutions for the subproblems are then simply propagated to all superproblems applying the inverse of the "renaming" to the solutions.

Example 2.17 (Structure Sharing)

Figure 2.2 *shows a graph where two subgraphs are shared.*

The conclusion drawn from the results of two implementations of this idea is that decomposition is a handy tool for theoretical issues of theory unification, especially general E-unification [GS89], but is not feasible for an efficient immediate implementation of a general equality reasoner. The main reason for this fact is that rewriting alone is more powerful than the decomposition mechanism combined with syntactic heuristics.

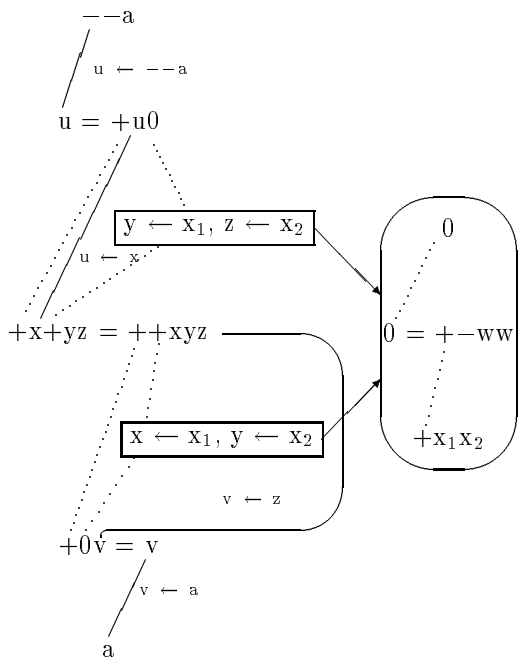
This does not exclude that for special domains semantic heuristics based on special knowledge can be very powerful as for example in the case of induction theorem proving [Hut90] (see also section 6.3).

We think that E-resolution and related methods work at the wrong level: if equations are present the replacement of equals is the main work in the proof and implicational steps which correspond to resolution have auxiliary and finishing character. Adequate calculi to handle resolution as a subcase of equational reasoning are presented in section 2.6. By the combination of E-resolution and E-unification the central problems are tried to be solved locally without any possibility to take the global context into account. Therefore decomposition is inadequate to handle equational problems that are mainly based on equality.

However, an approach related to decomposition can be used at a more general level to guide the search for refutation graphs as explained in chapter 4.

Figure 2.2: Structure Sharing

This is a variant of the fourth graph of example 2.16. The same (up to renaming) subproblem $0{=}{+}x_1x_2$ occurs at two different positions. The renaming substitutions are depicted in the boxes.



2.5 Rewriting

The observation that equations can be "applied" to terms led to a term replacement approach for the treatment of the equality relation. In order to obtain an algorithm which proves the equality of two terms, one can successively apply oriented equations to the terms. Such an algorithm only decides the equality of the terms but cannot make them equal by computing an instantiation of their variables as required for resolution based systems.

The main idea of a term rewriting system is to consider the equations as rules that may only be applied in one direction. The direction is determined by a partial ordering on the set of terms. A method to decide the equality of two terms under special equality theories can then be obtained by "reducing" the terms to a unique normal form using the directed equations. The theory axioms must obey certain conditions, namely, they must be confluent and Noetherian[(ix)], to ensure completeness and termination of the decision procedure. The equations defining the theory must be directable and must have the properties above or it must be possible to add other equations such that the new system is equivalent to the old one and has the desired properties. This procedure developed by D. Knuth and P. Bendix [KB70] is called completion. The system of directed equations constitutes a set of rewriting rules.

When computing a normal form all situations where two rules can be applied to derive different successors are potentially dangerous, because it must be ensured that both cases lead to the same normal form later on (confluence). D. Knuth and P. Bendix showed that it is sufficient to consider critical pairs just between the rules and to add the corresponding equations to ensure this property. Critical pairs can be constructed from two rules or two instances of the same rule if the left hand sides of the rules overlap, that means that some non-variable subterm of one left hand side can be unified with the other left hand side.

[(ix)] terminating

Definition 2.18 (Critical Pair Construction)

$$\begin{array}{l} l_1 \rightarrow r_1 \\ l_2 \rightarrow r_2 \\ +\hline \sigma(l_2[p \leftarrow r_1] = r_2) \end{array} \quad \textit{if} \quad p \textit{ is a non variable position of } l_2,\ \sigma \textit{ is an mgu of } l_2|p \textit{ and } l_1.$$

If we write $l \rightarrow r$ *we always assume that* $\sigma(l) > \sigma(r)$.

In principle the Knuth-Bendix completion algorithm then works as follows [KB70, HO80, Buc85, Der87, JL87]: beginning with a set of undirected equations, an empty set of directed rules, and a reduction ordering it tries to derive a confluent and terminating set of rules from the equations. It applies the following steps until no equations remain: Take an equation, apply all rules to the equation, direct the equation according to the given reduction ordering, and put it into the set of rules. Generate all critical pairs, that is, terms for which rule applications overlap, between the new rule and the set of rules and put them into the set of equations. If this algorithm terminates, it produces a set of rules that can be used to decide the equality of arbitrary sets of terms of the given theory.

A rule is applicable to a term if the left hand side of the rule matches the term or one of its subterms. If a rule is applied to an object with subterms to which it is applicable, then these are replaced by the right hand side of the rule with the matcher applied to it.

Definition 2.19 (Rewrite)

$$\begin{array}{l} l \rightarrow r \\ t \\ \downarrow\hline t[p \leftarrow \sigma(r)] \end{array} \quad \textit{if} \quad p \textit{ is a non variable position of } t \textit{ and } \sigma(l) = t|p.$$

In the field of automated deduction the application of the rules is often called demodulation [WRCS67, WOLB84] and we will use this term here too. In the field of automated deduction the orientation is often determined heuristically, unless an immediately obvious orientation of the equation is possible.

Meanwhile there are results in handling undirectable equations [BDP87, BDP89] and using special theory unification and matching algorithms [Sti85, KZ89] (see section 6.5). In addition there are attempts to use rewrite systems to construct universal unification algorithms [Kir87] and various methods using narrowing [Hul80]. Completion is also integrated with conditional equations [Ric83a, Pet83, Kap84b, ZR85, JW86, JL87, Rus87, ZK88, BG90] but unfortunately the unfailing as well as the conditional attempts are not as convincing as the pure method when unit equations are directable. Nevertheless almost all problems contain directable equations.

The techniques developed in the area of rewrite systems are widely used. Applications range from the simple and nevertheless powerful Knuth-Bendix equality reasoning procedure up to a special *rewriting logic* to describe the semantics of concurrency in programming languages [Mes90, Mes91].

Of course there are interrelations between unification theory and term rewriting systems and one goal is to combine rewriting techniques and unification algorithms.

Some results of the research in term rewriting systems led to universal unification algorithms restricted to theories with a confluent and terminating rewrite system. F. Fages [Fag83], J. Hullot [Hul80], J.-P. Jouannaud, C. Kirchner, H. Kirchner [JKK83], J. You, P. Subramayou [YS86], A. Martelli, C. Moiso, G. Rossi [MMR86], and C. Kirchner [Kir85] defined systems for this purpose.

Sometimes special theory unification algorithms are used in completion systems as a powerful instrument to handle unorientable equations. Such a method was used for example by M. Stickel [Sti85] to prove ring commutativity from $x^3 = x$. We shall come back to this point in section 6.5.

There are also interrelations between special deduction rules and rewriting systems; we shall discuss them in section 3.1.

The Knuth-Bendix algorithm itself is just a method to construct decision procedures for some equational theories. But unfortunately this type of decidable equational theory is very rare. What we need is a system that uses the benefits of this method for automated reasoning within a semidecision procedure. M. Bonacina and J. Hsiang [BH91] give an excellent framework for a unifying look at the various extensions of the completion procedure also including its value for automated theorem proving.

2.6 Restrictions for Paramodulation

The problem of unifying resolution, paramodulation, and rewriting was tackled by researchers of all the three areas. The goal is to obtain an automated reasoning calculus with the benefits of each of the areas, that is, the simplicity and efficiency of resolution, the availability of equations when formulating mathematical theories of paramodulation, and the ordering restrictions as well as the reduction power of rewriting.

Most people primarily concerned with resolution tend to separate the equational problem from the "main" problem. They circumvent the problem of conditional equations and reason that in most cases conditional equations occur together with unit equations which can be handled by completion and rewriting. Hence the solution of equational problems is seen as a preprocessing step of the resolution theorem prover. This viewpoint is for example a basis for most approaches that embed equations into Prolog (see page 53).

All advanced equality reasoning methods mentioned above are led astray when formulae like $A \Rightarrow x = y$ or $A \Rightarrow x = t$ (see example 4.1, page 39) are among the axioms. Such conditional equations are typical, however, for real situations and neither do they have any particular general structure that can be exploited for difference reduction nor are they directable, and there is no reason to believe that the "equality problem" is solved when a satisfactory procedure for handling the unit equations is found.

From the preprocessing idea above we obtain a complete theorem prover if we first generate all "good and necessary" rewrite rules and then use fair heuristics when applying the conditional equations using paramodulation.

In the present section we focus on a view of automated reasoning that imposes restrictions on the paramodulation inference rule. These restrictions are incorporated into calculi which consider resolution steps as special paramodulation steps and they have the property to behave like the Knuth-Bendix completion procedure if only unit equations are present. We name the calculi **superposition calculi** according to the name of their main inference rule.

In the following subsection we focus on the development of the superposition rule. We use the notions completion step, paramodulation step, and superposition step synonymously in this context. In chapter 4 we shall compare some of the paramodulation restriction strategies which we incorporated into the MKRP-system to obtain some heuristic criteria to decide when to use which strategy.

The development of the reduction operations can be considered separately from the development of the superposition rule. This is done in subsection 2.6.2.

2.6.1 Superposition

We consider the completion algorithm for conditional equations as a paramodulation calculus with restrictions on the application of the paramodulation rule. The calculus incorporates resolution as a special case of paramodulation. A resolution step between the literals L and $\neg L'$ for example is done by paramodulating $L = \mathit{True}$ into $L' = \mathit{False}$. Ordinary positive equations remain unaltered, negative equations can be written $(s = t) = \mathit{False}$.

This facet of paramodulation theorem proving leads to a homogeneous calculus regarding resolution and paramodulation as one rule. It completely eliminates the control problem of a paramodulation part grafted on a resolution theorem prover.

D. Lankford was the first who extended the concept of orientation of equations to paramodulation theorem proving [Lan75].

G. Peterson [Pet83] developed a resolution and paramodulation calculus which reduces to the Knuth-Bendix algorithm when only given unit equality axioms and theorems. G. Peterson's approach only allows for restricted reduction orderings and only demodulation by unit reduction rules. J. Hsiang and M. Rusinowitch [HR86, Rus87] extend the reduction ordering to literals, that is, only the "maximal" literals of the clauses must be considered for paramodulation. The restrictions on the orderings imposed by these works allow all commonly used orderings to be adapted. They use strong simplification orderings. H. Zhang and D. Kapur [Zha88, ZK88] extended it to more orderings and contextual rewriting.

We implemented some superposition rules as strategies in the MKRP-system. The implementation and comparison (see chapter 4) makes it necessary to have the rules available in the same notation. Hence we present the rules not in the original notation but instead in the notation as introduced above, also to gain a homogeneous presentation. We omit the factoring rule which is always necessary to get a complete calculus, as in the case of resolution. The differences in the definitions 2.20 through 2.23 are emphasized in the print, because the rules themselves are very similar.

In the definition 2.20 of J. Hsiang and M. Rusinowitch superpositions with the left and right hand side of an equation into a maximal literal are allowed. The problem of ordered paramodulation is that it does not reduce to Knuth-Bendix completion in the case when only unit equations are present. For this situation ordered paramodulation is weaker than completion, in the sense that it has a larger search space.

In the following definitions a Literal L is called *maximal with respect to* a set of literals $\{L_1, \ldots, L_n\}$ iff for all i $L_i \not\succ L$, we call it *strictly maximal* if $L_i \not\succ L$ and $L_i \neq L$, hence does not occur in the set (see [BG90, page 430]).

Definition 2.20 (Ordered Paramodulation [Rus87, page 49])

$$\frac{\begin{array}{l} s = t, L_1, \ldots, L_n \\ l = r, K_1, \ldots, K_m \end{array}}{+\ \sigma(s[p \leftarrow r] = t, L_1, \ldots, L_n, K_1, \ldots, K_m)} \quad \textit{if}$$

p is a non variable position of s, σ is an mgu of $s|p$ and l, $s = t$ is maximal with respect to $\{L_1, \ldots, L_n\}$, **$l = r$ is maximal with respect to $\{K_1, \ldots, K_n\}$,** *and $\sigma(r) \not\succ \sigma(l)$.*

In the definition 2.21 of M. Rusinowitch superposition into left hand sides of all literals is allowed. This rule reduces to the critical pair creation of D. Knuth and P. Bendix, but is weaker than ordered paramodulation in the conditional case where more operations are allowed.

Definition 2.21 (Weak Clausal Superposition [Rus91, page 24])

$$\frac{\begin{array}{l} s = t, L_1, \ldots, L_n \\ l = r, K_1, \ldots, K_m \end{array}}{+\ \sigma(s[p \leftarrow r] = t, L_1, \ldots, L_n, K_1, \ldots, K_m)} \quad \textit{if}$$

p is a non variable position of s, σ is an mgu of $s|p$ and l, $\sigma(s = t)$ is maximal with respect to $\{\sigma(L_1), \ldots, \sigma(L_n)\}$, **$\sigma(t) \not\succ \sigma(s)$,** *and $\sigma(r) \not\succ \sigma(l)$.*

H. Zhang's and D. Kapur's inference rule 2.22 [ZK88] removes both restrictions, that is, it possesses the property to reduce to Knuth-Bendix completion as well as the constraint to consider only maximal literals.

Definition 2.22 (Strict Clausal Superposition, [ZK88, page 7])

$$\frac{s = t, L_1, \ldots, L_n \quad l = r, K_1, \ldots, K_m}{+\ \sigma(s[p \leftarrow r] = t, L_1, \ldots, L_n, K_1, \ldots, K_m)} \quad \textit{if}$$

p is a non variable position of s, σ is an mgu of $s|p$ and l, $s = t$ is maximal with respect to $\{L_1, \ldots, L_n\}$, $\boldsymbol{l = r}$ **is maximal with respect to** $\boldsymbol{\{K_1, \ldots, K_n\}}$, $\boldsymbol{t \not> s}$, *and $r \not> l$.*

It is somewhat dubious that the condition is formulated without the substitution σ. Of course it is possible to comprise the substitution in the inference rule. Unfortunately H. Zhang's refutation system is not complete together with the tautology removal rule, as L. Bachmair and H. Ganzinger showed with an example [BG90]. They also propose how the system can be repaired such that this incompleteness does not longer occur. They defined the following refined strict superposition rule and additionally a merging paramodulation rule (we call it merging superposition).

Definition 2.23 (Strict Superposition, [BG90, page 431])

$$\frac{sign^{(xi)}(s = t), L_1, \ldots, L_n \quad l = r, K_1, \ldots, K_m}{+\ \sigma(sign(s[p \leftarrow r] = t), L_1, \ldots, L_n, K_1, \ldots, K_m)} \quad \textit{if}$$

- *p is a non variable position of s, σ is an mgu of $s|p$ and l*
- *$\sigma(r) \not> \sigma(l)$*
- *$\sigma(l = r)$ strictly maximal with respect to $\sigma(K_1, \ldots, K_m)$*
- *$\sigma(l)$ does not occur in the negative literals of $\sigma(K_1, \ldots, K_m)$*
- *$\sigma(t) \not> \sigma(s)$*
- *$\sigma(sign(s = t))$ is strictly maximal with respect to all positive literals of $\sigma(L_1, \ldots, L_n)$. If sign is positive it is also strictly maximal with respect to all negative literals, else it is only maximal*

Compared with H. Zhang's rule strict superposition has two refinements beside the consideration of the substitution, both sharpening the restriction on the applicability:

1. $\sigma(l)$ does not occur in the negative literals of $\sigma(K_1, \ldots, K_m)$.
2. The notion "strictly maximal".

The merging inference rule must be added to circumvent the tautology problem of H. Zhang's calculus. It is a genuine superposition into a right hand side of a directable equation under certain restrictions. Two similar (they have the same left hand side) positive literals in a clause are made equal by successive applications of merging superposition and one final factoring step, thus removing the literal causing the possibly necessary tautology in H. Zhang's inference system.

$^{(xi)}$ With $sign$ we denote that this literal is positive (not negated) or negative (negated with $\neg$).

Definition 2.24 (Merging Superposition, [BG90, page 432])

$$\frac{\begin{array}{l} s = t, s' = t', L_1, \ldots, L_n \\ l = r, K_1, \ldots, K_m \end{array}}{+\quad \sigma(s = t[p \leftarrow r], s = t', L_1, \ldots, L_n, K_1, \ldots, K_m)} \quad \textit{if}$$

- $\sigma = \tau\rho$ *where* τ *is an mgu of* $t|p$ *and* l, ρ *an mgu of* $\tau(s)$ *and* $\tau(s')$, *and* p *is a non variable position of* t
- $\sigma(r) \not> \sigma(l)$
- $\sigma(l = r)$ *strictly maximal with respect to* $\sigma(K_1, \ldots, K_m)$
- $\sigma(l)$ *does not occur in the negative literals of* $\sigma(K_1, \ldots, K_m)$
- $\sigma(s) \not> \sigma(t)$
- $\sigma(s = t)$ *strictly maximal with respect to* $\sigma(L_1, \ldots, L_n)$
- $\sigma(s)$ *does not occur in the negative literals of* $\sigma(L_1, \ldots, L_n)$
- $\tau(s) > \tau(t)$ *and* $\sigma(t') \not\geq \sigma(t)$

Using any one of these superposition methods drastically changes the resolution strategy because it can only be resolved or paramodulated on links joining maximal literals. In this case no set-of-support or linear strategy is complete even for resolution alone.

The invention and distribution of Prolog showed that it is very promising to consider special rule systems for Horn clauses. The first who studied the restriction of superposition to Horn clauses was E. Paul [Pau85] but his algorithm only works for orientable equations.

N. Dershowitz [Der91] presented two restriction strategies for paramodulation, specialized for Horn clauses, which reduce to Knuth-Bendix completion because they are refinements of strict superposition. The first one is a unit strategy with the two rules of definition 2.25. The second one is called a decreasing strategy and we do not elaborate on it (the description in [Der91] is not detailed enough).

Definition 2.25 (Unit Superposition, Unit Narrowing [Der91, page 120])

$$\frac{\begin{array}{l} s = t \\ l = r \end{array}}{+\quad \sigma(s[p \leftarrow r] = t)} \quad \textit{if} \quad p \textit{ is a non variable position of } s, \sigma \textit{ is an mgu of } s|p \textit{ and } l, \sigma t \not> \sigma s \textit{ and } \sigma r \not> \sigma l^{(xiii)}.$$

$$\frac{\begin{array}{l} s \neq t, \neg L_1, \ldots, \neg L_n \\ l = r \end{array}}{+\quad \sigma(s[p \leftarrow r] \neq t, \neg L_1, \ldots, \neg L_n)} \quad \textit{if} \quad p \textit{ is a non variable position of } s, \sigma \textit{ is an mgu of } s|p \textit{ and } l, \sigma L_1, \ldots, \sigma L_n, \sigma t, \sigma(s[p \leftarrow r]) \not> \sigma s.$$

There are other approaches to restrict paramodulation in the case of Horn clauses. M. Rusinowitch for example gives a refinement of his calculus to Horn clauses [Rus91, section 7]. U. Furbach, S. Hölldobler, and J. Schreiber propose linear paramodulation as a natural extension of the SLD strategy [FHS91]. Unfortunately linear paramodulation requires paramodulation with variables.

W. Snyder and C. Lynch proposed a restriction based on basic narrowing, thus inhibiting certain positions for paramodulation steps. In particular they use a constructive method for the completeness proof of this calculus providing the possibility to see the effects of further restrictions to completeness.

(xiii) The last condition was originally formulated as $\sigma(s[p \leftarrow r]) \not> \sigma s$.

Definition 2.26 (Basic Paramodulation [SL91a, Definition 3.5])

$$\frac{sign(s = t), L_1, \ldots, L_n \quad l = r, K_1, \ldots, K_m}{+\ \sigma(sign(s[p \leftarrow r] = t), L_1, \ldots, L_n, K_1, \ldots, K_m)}$$

if

- *p is a non variable position of s, σ is an mgu of $s|p$ and l*
- $\sigma(r) \not\succeq \sigma(l)$
- *$\sigma(l = r)$ maximal in $\sigma(K_1, \ldots, K_m)$*
- $\sigma(l = r) \not\succeq \sigma(s = t)$
- *$\sigma(s = t)$ is maximal with respect to $\sigma(L_1, \ldots, L_n)$*
- *the root of $s|p$ is unmarked*
- *a mark is placed on each position in the conclusion introduced by instantiation via σ*
- *a mark is placed on each position of $\sigma(r)$ in the substitute*

It is clear that any kind of rewrite reduction must be adapted to handle the marks placed by the basic paramodulation rule. Unfortunately the reduction operations (rewriting as well as subsumption) are incomplete together with this rule. In an extension of the calculus L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder developed restrictions for the reduction rules that lead to a complete inference system [BGLS92].

As mentioned above we incorporated some of the cited paramodulation restriction strategies into our theorem prover and shall give an evaluation in section 4.2.1.

2.6.2 Reduction

The power of Knuth-Bendix completion for theorem proving heavily relies on the strong reduction facility. In parallel to the extensions of the critical pair construction to clauses the rewrite operation was extended to conditional rewrite rules. We give a shorter overview on reduction rules because we did not concentrate on different reduction rules in our implementation but just on suitable extensions of the MKRP reduction facility (see section 3.3). The definitions of D. Brand, J. Darringer, and W. Joyner [BDJ79], D. Lankford [Lan79], J.-L. Rémy [Rém82], and S. Kaplan [Kap84a] cleared the way for the more general contextual rewriting ones given by H. Ganzinger [Gan91] and H. Zhang [ZK88] (see below). L. Bachmair and H. Ganzinger [BG90] finally extended it to the semantically induced notion of redundancy which encompasses tautology deletion, subsumption, contextual rewriting, and elimination of redundant atoms (literals).

In this section we are in less detail than for the superposition rules above, because we think that for reduction it is essential to detect potential applications efficiently in a concrete context and not how powerful the reduction rule itself is defined. Therefore we just cite H. Zhang's very general rule [ZK88], which reduces to Knuth-Bendix reduction in the case of unit equations, and which can also be used to apply the subsumption rule.

Definition 2.27 (Contextual Rewriting [ZK88, page 12])

$$\frac{l \rightarrow r,\ K_1,\ \ldots,\ K_m \quad L_0,\ L_1,\ \ldots,\ L_n}{\downarrow\ L_0[p \leftarrow t'[p' \leftarrow \sigma(r)]],\ L_1,\ \ldots,\ L_n}$$

if

- *p is a non variable position of L_0*
- $C = \{\neg L_1, \ldots, \neg L_n\}$
- $L_0|p \cong_C t'$
- *p' is a non variable position of t'*
- $\sigma(l) = t'|p'$
- $\forall i : \sigma(\neg K_i) \rightarrow_C True$

This definition needs some explanations: $t'|p'$ can be reduced to $\sigma(r)$ in the context C. The context can be used to reduce the conditions $\neg K_1, \ldots, \neg K_m$ of the conditional rewrite rule $l \rightarrow r,\ K_1, \ldots, K_m$ to *True*. $\cong_C$ (constant congruence) is the minimal congruence relation which contains all equations of C ($\{\neg L_1, \ldots, \neg L_n\}$). For full equality the substitution property must be satisfied in addition.

Example 2.28 (Contextual Rewriting, taken from [ZK88])

$HasTeeth(x) \rightarrow True \vee \neg IsTiger(x)$

$\neg HasTeeth(y) \vee \neg IsTiger(y)$

We can reduce $\neg HasTeeth(y)$ *in the second clause to* $\neg True$ *and hence to False and remove the literal, because with the context* $C = \{IsTiger(y)\}$ *the condition* $IsTiger(x)$ *for* $HasTeeth(x) \rightarrow True$ *can be reduced by the context literal* $IsTiger(y) \rightarrow True$ *to True.*

Example 2.29 (Contextual Rewriting with Equational Conditions)

$a \rightarrow b \vee f(c) \neq f(d)$

$P(a) \vee c \neq d$

We can reduce $P(a)$ *to* $P(b)$ *in the second clause because with the context* $C = \{c = d\}$ *the condition* $f(c) = f(d)$ *can be reduced to True. The logical reason is as follows:*

$$
\begin{array}{lllll}
 & c = d & \Rightarrow & P(a) & \\
\textit{hence} & c = d & \Rightarrow & P(a) \wedge f(c) = f(d) & \textit{because } f \textit{ is a function} \\
 & & \Rightarrow & P(a) \wedge a = b & \textit{because } f(c) = f(d) \Rightarrow a = b \\
 & & \Rightarrow & P(b) & \\
\textit{that is} & c = d & \Rightarrow & P(b) &
\end{array}
$$

2.6.3 Summary

The extensions of completion to all versions of superposition calculi are very powerful because of the constraints on the paramodulation rule and the reduction facility. In addition every resolution and paramodulation theorem prover can be controlled in such a way that it simulates this calculus and all features of the underlying prover remain available in the absence of equations (see chapter 3). This result rests on the fact that resolution is a subcase of paramodulation in the usual presentation of superposition calculi. The empirical results in chapter 4 and in appendix B corroborate our view, namely, that choosing paramodulation with restrictions as a basis of the calculus is the adequate way to build equality theorem provers.

2.7 Narrowing

Up to now we considered difference reduction and superposition as two completely incompatible approaches to equality reasoning. In this section we shall discuss a special technique developed in unification theory that can serve as a basis to unify both approaches.

After the discovery of confluent and terminating rewrite systems, attempts were made to use them in a broader context than simple reduction, such that not only the equality of terms could be decided but also proper solutions for equational problems could be computed. In order to do so, the matching in the directed application of equations for rewriting is replaced by complete unification. Of course this is not a reduction operation and hence the complete search space must be taken into account, which is nevertheless considerably reduced by the orientation of the equations and further constraints on the positions to be considered. For this constrained usage of equations the notion "narrowing" was moulded.

The unification algorithm given in definition 2.15 can be extended to the following one using narrowing.

Definition 2.30 (E-Unification Using Narrowing)
Given a set R constituting a confluent and terminating rewrite system we obtain a complete and correct R-unification procedure by adding a fifth rule to the algorithm in definition 2.15.

1. *Switching: replace an equation $t = x$ by $x = t$.*
2. *Deletion: delete $t = t$.*
3. *Decomposition: replace $fs_1 \ldots s_n = ft_1 \ldots t_n$ by $s_1 = t_1, \ldots, s_n = t_n$.*
4. *Elimination: if $x = t$ is an equation where x does not occur in t, then replace all occurrences of x by t in all other equations.*
5. *Narrow: replace $t = s$ $(s = t)$ by $\sigma(t[p \leftarrow r]) = \sigma(s)$ if p is a non variable position of t, $l \rightarrow r \in R$, and $\sigma(l) = \sigma(t|p)$.*

Step 5 is a branching step and all possibilities to apply this step must be considered to obtain all unifiers.

Narrowing is an adequate tool to perform E-unification relative to a confluent and terminating theory interleaved with other steps. Of course narrowing can be used before this confluent and terminating set of rewrite rules is derived. It can be used even if no such set exists but then only in a heuristical manner.

Hence narrowing can serve to perform difference reduction, that is, to derive E-resolution steps. It can be controlled in a lazy way, that is, the incorporation of new rewrite rules is allowed. Hence with lazy we denote another thing than usually in the context of narrowing, where it roughly models "call-by-need" [Red85].

M. Fay [Fay79] and J. Hullot [Hul80] were the first to use narrowing techniques to construct unification algorithms for a confluent and terminating theory, J. Hullot found further constraints and called his procedure basic narrowing. Basic means that positions which are prefixes of the narrow position must not be considered for further steps.

L. Fribourg [Fri84a, Fri85b, Fri85a] made additional restrictions, he used an innermost strategy, but his method requires more conditions to be fulfilled for the equational theory.

P. Bosco, E. Giovannetti, and C. Moiso [BGM88] give a better strategy developed from SLD-resolution. They introduce non terminating rewrite systems which are used for modelling infinite datastructures. For this method normalizing on the intermediate results is not possible (selection narrowing). Their paper also shows how to represent equational problems only at the predicative level. P. Bosco, C. Cecchi, and C. Moiso incorporated this approach into the Prolog compilation technique [BCM89].

M. Fay [Fay79] as well as P. Rety, C. Kirchner, H. Kirchner, and P. Lescanne [RKKL85] use normalizing narrowing, which allows to keep intermediate results reduced. The second work introduces subsumption of newly derived terms and a finite representation of loops in the narrowing tree.

A good overview of these different strategies of narrowing is given by S. Krischer [Kri90]. Statistics about the effect of the various restrictions are given which demonstrate their respective usefulness.

There are various areas where narrowing can be used. For example, it can serve as an operational model of functional languages. This technique is used in the context of logic programming [Red85, DG89].

Another extension is that to a universal unification algorithm for conditional theories [Hus85].

The various restriction strategies developed for narrowing ask for an extension to general paramodulation. W. Snyder and C. Lynch [SL91a] give two such restrictions on paramodulation corresponding to the basic strategy (see also definition 2.26). Unfortunately these restrictions do not lead to a complete inference system together with rewrite reductions.

As in the case of rewriting where L. Wos developed a heuristic version, he and his fellow researchers were successful to develop a technique of combining several equality reasoning steps to a heuristic extension of paramodulation.

Hyperparamodulation was defined by L. Wos, R. Overbeek, and L. Henschen [WOH80] as follows:

Definition 2.31 (Hyperparamodulation)

$s_1 = t_1$
$\vdots$
$s_n = t_n$
C_0 *if* $t'_1, \ldots, t'_n$ *are distinct terms in* C_0 *and* $\sigma(t'_i) = \sigma(t_i)$ *for* $i \in \{0, \ldots, n-1\}$
$\vdots$
$+$———————
C_n

Then C_n *is a* **hyperparamodulant** *with nucleus* C_0 *and satellites* $s_1 = t_1, \ldots, s_n = t_n$.

It is clear that hyperparamodulation is complete in the sense that all paramodulation steps are hyperparamodulation steps with $n = 1$. It is a straightforward restriction for hyperparamodulation to apply it just in those cases where a new resolution possibility is introduced by the successive paramodulations. In chapter 3 we shall give a rule with the aim to combine narrowing and difference reduction.

Chapter 3

Variations of the Basic Superposition Calculus

In the previous chapter we presented various equality reasoning methods which we wanted to combine into one single system. In this chapter we describe the corresponding extensions of the calculus of the Markgraf Karl system and clarify their influence on the completeness of the overall system.

In section 1.2 we motivated extensions of the Knuth-Bendix algorithm as a basis for an equality reasoning procedure. For conditional equations this amounts to a superposition approach. In the sequel we shall show how this mechanism can be embedded into a system based on resolution in general and on clause graph resolution in particular.

We extend it by more elaborated inference and reduction rules for clause graph resolution. The handling of special methods such as narrowing is incorporated into the superposition calculus through the definition of abbreviation inference rules. Narrowing will be used to make theory resolution steps practically feasible. We shall elaborate in section 4.1 on a technique to perform the narrowing steps only if they are useful at a higher level, that is, if a refutation graph structure can be detected in the clause graph or if their application may lead to such a structure.

3.1 Inference Rules

Each of the inference rule systems given in section 2.6 can be used as a restriction strategy for the paramodulation and resolution rules. As default we propose to use L. Bachmair's and H. Ganzinger's approach (definitions 2.23 and 2.24) because of its stronger restrictions. For the narrowing technique presented in 3.1.1 it is not important which of the available restrictions of superposition is chosen.

The superposition steps comprise selected resolution and paramodulation steps. Example 3.1 shows the correspondence of a resolution step as defined in definition 2.9 to a superposition step.

Example 3.1 (Resolution Step as Superposition)
Superposition is combined with the deletion of a literal "True=False":

$$\frac{\begin{array}{l} L_0 \rightarrow \mathit{True},\ L_1,\ \ldots,\ L_n \\ K_0 \rightarrow \mathit{False},\ K_1,\ \ldots,\ K_m \end{array}}{+\ \sigma(L_1,\ \ldots,\ L_n,\ K_1,\ \ldots,\ K_m)} \quad \textit{if} \quad \sigma \textit{ is an mgu of } L_0 \textit{ and } K_0.$$

It is clear that superposition on the maximal literals corresponds to a calculus which does not necessarily find a proof which is as short as possible. But the restriction is strong enough to lead to a much reduced search time, even if it results in a considerably longer proof. Especially if only proper paramodulation steps occur, that is no resolution steps, this restriction is very useful.

3.1.1 E-Resolution via Narrowing

In order to integrate a difference reduction mechanism into the basic calculus we shall use narrowing (see section 2.7).

In the sequel we describe how E-resolution steps can be performed as special hypersteps which combine arbitrary many narrowing steps. With this technique the inference system is enhanced concerning directable unit equations. For this purpose we use a method which is very simple and general but nevertheless can be extremely powerful: we allow additional steps in the calculus which are just useful abbreviations. Using such a technique the completeness of the basic superposition calculus is retained and only the soundness of the abbreviations must be proven. In our case the abbreviations are always combinations of correct steps and hence are also correct.

We only concentrate on one abbreviation rule in this thesis: one E-resolution step abbreviates several paramodulations represented by the narrowing steps. Another candidate for the abbreviation technique would be induction.

A single superposition step generates a clause with one paramodulated literal and any number of instantiated literals. The maximal literal of the new clause need not necessarily be the successor of the maximal literal of the parent clause (see figure 3.1). Hence it is not possible in general to apply sequences of superposition steps to one literal in order to make it resolvable within the pure superposition calculus. But following such a linear strategy is desirable because linear strategies are working according to an implicit plan, namely to remove successively all literals of one clause. Hence they tend to solve the subproblems defined by the literals separately. Of course they are more endangered to be led astray in the depths of the search space.

Figure 3.1: Example Maximal Literals

$Pfx \vee Px$	Descendent of Pfx:	Pb
|	Descendent of Px:	Pa
	Ordering:	$a > f > b$
$fa \rightarrow b$	Maximal literal of $Pfx \vee Px$:	Pfx
Paramodulant: $Pb \vee Pa$	Maximal literal of $Pb \vee Pa$:	Pa

The operation of making two literals resolvable is just general E-unification and can be performed using narrowing. The narrowing steps can be controlled such that they reduce the differences (see section 2.4) of the literals. We define narrowing steps to successively produce theory unifiers in the following way:

Definition 3.2 (Narrowing Inference)

$$\frac{\begin{array}{l} L_0^{t^1 \ldots t^k},\ L_1,\ \ldots,\ L_n \\ l \rightarrow r \\ \downarrow \end{array}}{L_0^{t^1 \ldots t^k t^{k+1}},\ L_1,\ \ldots,\ L_n} \quad \textit{if}$$

- *each* t^i *is a pair* (L^i, σ^i)
- L^i *is a literal derived from* L_0 *by successive applications of equations and reductions*
- σ^i *is the merge of the unifiers of these applications and reductions*
- L^{k+1} *is the normalized* $\sigma(L^j[p \leftarrow r])$
- σ^{k+1} *is the corresponding extended unifier*
- p *is a narrowing position in* L^j
- $\sigma(L^j|p) = \sigma(l)$

The narrowing process can be depicted using a tree like that in figure 3.2. The string $t^1 \ldots t^k$ of the definition above is used to record the nodes of the narrowing tree.

Example 3.3 (Narrowing Tree as a String)

Figure 3.2 *shows one of the successful paths through the narrowing tree leading to the first E-resolution step of example* B.3.2[(i)]*. The derived unifier is* $\{y \leftarrow c + (-b),\ x \leftarrow a\}$.

The depicted tree in the notation $t^1 \ldots t^k$ *of the narrowing inference rule is:*

$(y \cdot x \neq 0, \{\})$

$(z' \cdot y' + x' \cdot y' \neq 0, \{y \leftarrow z' + x', x \leftarrow y'\})$

$(b \cdot a + x'' \cdot a \neq 0, \{y \leftarrow c + x'', x \leftarrow a\})$

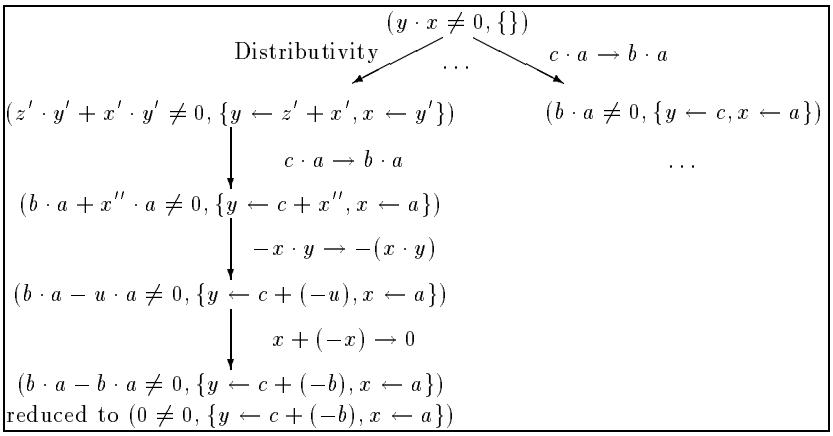
$(b \cdot a - u \cdot a \neq 0, \{y \leftarrow c + (-u), x \leftarrow a\})$

$(0 \neq 0, \{y \leftarrow c + (-b), x \leftarrow a\})$

...

$(b \cdot a \neq 0, \{y \leftarrow c, x \leftarrow a\})$

Figure 3.2: Narrowing Tree



In the string $t^1 \ldots t^k$ the narrowing tree of L_0 is recorded including all results after normalization. The intermediate results would be only necessary to obtain a better understandable proof, but they could be handled in the same manner as the narrowing steps if this is desired. The process of producing t^{k+1} can be controlled in any way proposed by one of the known narrowing strategies.

The information collected by the narrowing process is used to perform E-resolution steps. The strategy is to produce resolvable literals via narrowing and then to E-resolve upon them. When a reflexivity clause $x = x$ is present NRF-like steps[(ii)] are not necessary.

Definition 3.4 (E-Resolution Inference)

$$\frac{L_0^{t^1 \ldots t^k},\ L_1,\ \ldots,\ L_n \qquad K_0^{s^1 \ldots s^l},\ K_1,\ \ldots,\ K_m}{+\ \rho(\tau^i(L_1,\ \ldots,\ L_n),\ \sigma^j(K_1,\ \ldots,\ K_m))} \quad \textit{if}$$

L_0 and K_0 have same predicate symbols and opposite signs and there exists an mgu ρ of L^i and K^j in $t^i = (L^i, \tau^i)$ and $s^j = (K^j, \sigma^j)$.

The way how E-resolution is integrated into the calculus can be regarded as a scheme of how other specialized inference rules – as for example an induction step – could be handled. Another rule to abbreviate the forward search induced by the completion procedure by considering a goal is to incorporate a "forward closure" rule into the calculus [SA92].

(i) Resulting in `R243`.

(ii) See definition 2.14.

3.1.2 Completeness Considerations

The soundness of the inference system with E-resolution follows immediately because the new E-resolution rule is just a combination of several paramodulation steps. Completeness is also obvious because all paramodulation steps remain possible as before.

Of course it is interesting to consider potential restrictions on the superposition steps. We distinguish three cases where the calculus remains complete:

- If equations occur only as units and constitute a confluent and terminating rewrite system, narrowing and E-resolution alone are possible, and we can get rid of the paramodulation rule. Then the inference system realizes a lazy E-unification process via narrowing, and hence it is complete if E-resolution and narrowing steps are selected in a fair way. Theory resolution with the specified equational theory is performed. For a proof that narrowing in general constitutes a complete unification algorithm for a confluent and terminating theory with an infinite set of unifiers see for example M. Fay [Fay79].

- If equations only occur as units and do not constitute a confluent and terminating rewrite system, narrowing and E-resolution may become complete without superposition during the deduction process. If we start with an inference system using paramodulation, narrowing, and E-resolution and a confluent and terminating rewrite system is derived using the given reduction ordering, then we can proceed as follows: all paramodulants but the unit equations of the confluent and terminating system are removed from the clause set and from now on only narrowing and E-resolution, and no paramodulation inferences are performed, that is, the theorem prover is restarted with new input and then behaves as in the first case.

- If arbitrary equations with conditions occur, narrowing and E-resolution are just used as additives to the basic superposition calculus. The E-resolution steps are abbreviations for sequences of paramodulation steps which could still be executed separately.

  The key problem in this case is how to use the lemmata, that is, the derived equations, in order to perform the narrowing steps. Because of the constraining effect of the narrowing strategies we propose lazy narrowing in the sense that new narrowing rules can be inserted into the narrowing process and not only additional problems as usual. In section 4.1.3 we shall discuss the control of interleaving narrowing and superposition steps.

The central question for further research is what may be done with the intermediate results. Such a restriction seems to be: all steps represented in the tree can be disregarded for superposition when they cannot contribute to the inference of new equations. In such cases nothing can be done with the intermediate results.

3.2 Restriction Strategies

Now we come to the problem of incorporating superposition techniques into our theorem prover. For the incorporation of equality reasoning into an existing theorem prover it is advantageous to regard a superposition calculus as a strategy to control the resolution and paramodulation rules.

There are well-known resolution strategies which are not complete, for instance unit or input resolution, but nevertheless they are useful to solve problems that practically cannot be solved with other strategies. This is a motivation to also use incomplete paramodulation strategies.

There are special tools for resolution theorem provers, for example the terminator [AO83] (see also section 6.4). To retain the power of such dedicated mechanisms justifies incomplete equality handling,

especially when it is complete for the case where only unit equations occur. This is the aim of the clause graph strategy in section 3.2.2.

Additionally it need not always be the case that a strongly constraining and restricting strategy is the best one, regardless of its completeness, because it could produce long proofs with many auxiliary results.

To remain complete we introduced the E-resolution rule only as a short cut for a combination of several paramodulation steps, but of course it can also be used as an incomplete restriction strategy only allowing E-resolution steps.

3.2.1 The Superposition Strategy

We presented various superposition rules in section 2.6. Here we just mention some aspects of their implicit resolution possibilities.

The usual superposition strategy makes no distinction between axiom and theorem clauses, because the set-of-support strategy is incomplete in this case. This is not important in most equality reasoning problems because a forward reasoning method is adequate, but it is essential when using resolution. M. Bonacina and J. Hsiang [BH91, section 5] as well as R. Socher-Ambrosius [SA92] elaborated the problem of goal oriented completion.

In the context of a different search behaviour of theorem provers and a view of multiple agents [Den91] it is essential to incorporate more than one restriction[iii] or selection strategy into such a system (see also the discussion in chapter 4). Of course ordered resolution is not compatible with the usual resolution strategies like linear resolution and set-of-support. An example for the incompleteness of the second one is the set of clauses $\{P, \neg P \vee \neg Q, Q\}$ with $P > Q$, where Q is the set-of-support.

3.2.2 A Clause Graph Strategy

As mentioned several times in this thesis the MKRP-system is based on the clause graph resolution calculus which was first proposed by R. Kowalski [Kow75]. Its main advantage is the inheritance mechanism for resolution possibilities. For a short introduction into the calculus see section 2.1. N. Eisinger proved that the clause graph calculus itself is refutation complete and refutation sound [Eis88, theorems 3.3_2 and 3.3_3]. But a calculus must always be regarded together with a strategy to select the next applicable step. As argued by N. Eisinger [Eis88, page 162ff] the clause graph calculus can get into cycles, even with a fair control strategy.

However, the experience of our research group with the clause graph procedure shows that in practice completeness is not the problem.

We like to embed equality reasoning into the clause graph procedure. With "embed" we mean that the calculus behaves as before if no equations are among the input formulae and hence no equational operations occur. We introduce the notion "as complete as", because we are not interested in the intrinsics of the clause graph calculus, we just examine the effect of the additional paramodulation rule.

Definition 3.5 (as Complete as)
Let A be a set of clauses and let $\mathcal{E}(A)$ be the set A together with the equality axioms corresponding to definition 1.1.

A calculus $\mathcal{K}_1$ is **as complete as** *$\mathcal{K}_2$ iff*

- *Whenever the empty clause is derivable in $\mathcal{K}_1$ from A then it is also derivable in $\mathcal{K}_2$ from A.*

[iii] J. Denzinger only uses different selection strategies because of the difficulties to value the results obtained with different orderings: a "good rule" in the one system can be a "bad equation" in the other one.

– *Whenever the empty clause is derivable from $\mathcal{E}(A)$ in $\mathcal{K}_1$ then it is derivable from A in $\mathcal{K}_2$.*

The projected procedure would be to allow paramodulation steps wherever possible, to reconstruct all links to paramodulated literals, and to inherit R-links to all other new literals, but first we try to clarify the influence of the ordering restriction on the clause graph procedure.

With ordered clause graphs we denote usual clause graphs with a restriction on the allowed resolution links, corresponding to the superposition rule.

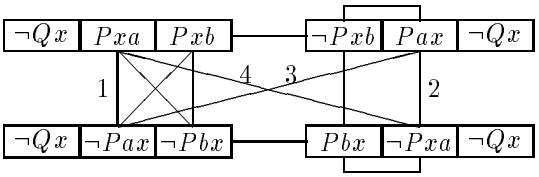
Theorem 3.6 (Ordered Clause Graphs) *Ordered clause graph resolution with the following link inheritance and construction rules is incomplete.*

– *Delete resolution link.*

– *Inherit all R-links as described in [Eis88].*

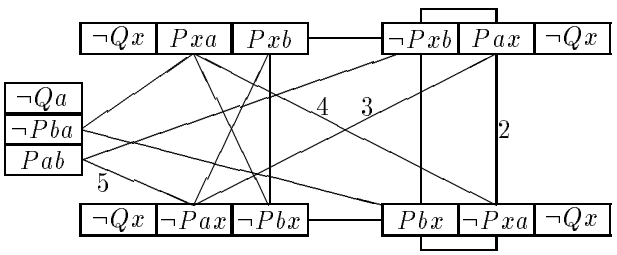
– *Removal of tautologies and subsumed clauses.*

Proof: The counterexample is taken from [Eis88][(iv)]. There it is used to show the inadequacy of the class of lifting lemmata usually proposed to prove the completeness. The links allowed in L. Bachmair's and H. Ganzinger's strategy (see definitions 2.23 and 2.24) are always labeled with a number, that is, only links with numbers represent possible resolution steps.



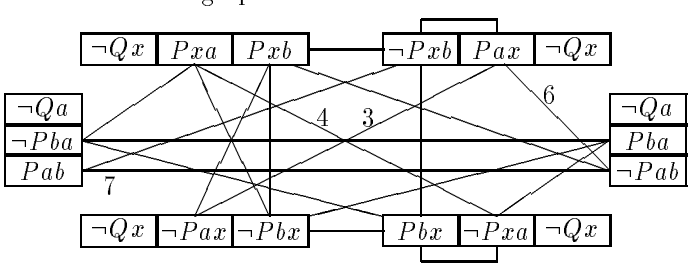
Together with the clause $Qa \vee Qb$ the specified set of clauses is unsatisfiable. In the following figures we omit the clause $Qa \vee Qb$ and all its incident links as long as they do not contribute to the deduction.

After resolution on link 1 we have the following graph.



Resolution on 5 leads to $R_5 : \neg Qa \vee \neg Pba \vee \neg Qb \vee \neg Pbb$, which we do not depict.

Resolution on link 2 leads to the next graph.

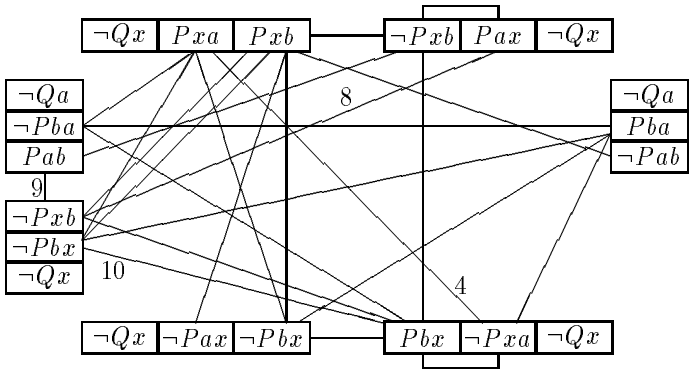


Resolution on 6 leads to $R_6 : \neg Qb \vee \neg Pbb \vee \neg Qa \vee Pba$, which we do not depict.

[(iv)] Unfortunately we could not construct a smaller example.

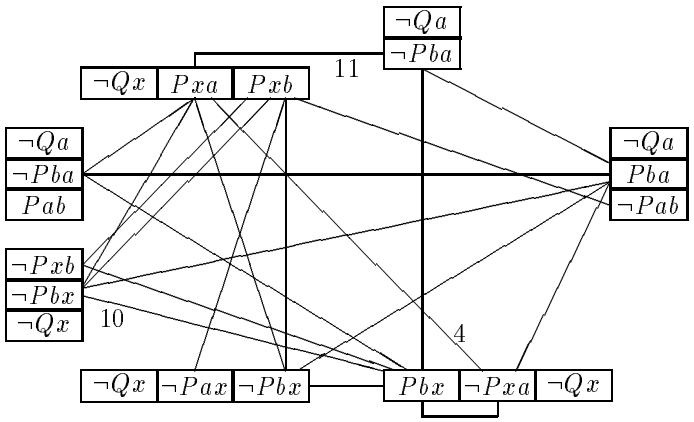
Resolution on 7 leads to $R_7 : \neg Qa \vee \neg Pba \vee \neg Qa \vee Pba$, which we do not depict.

After resolution on 3 we obtain the following graph.



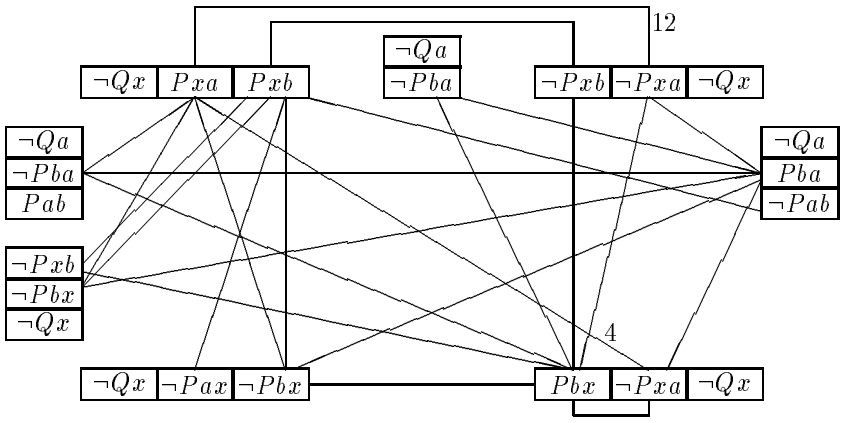
Resolution on 8 leads to $R_8 : \neg Qb \vee \neg Pbb \vee \neg Qa \vee \neg Pba$, which we do not depict. After this operation the upper right input clause is pure and can be deleted.

Resolution on 9 leads to the following graph.

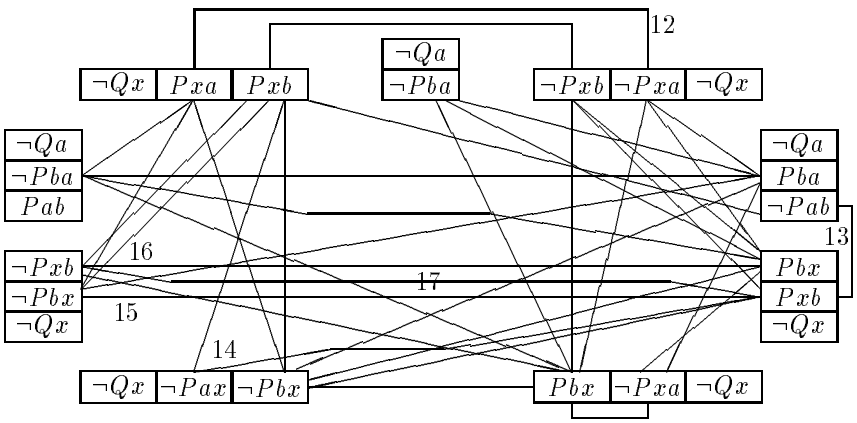


Resolution on 11 leads to $R_{11} : \neg Qb \vee Pbb \vee \neg Qa$, which we do not depict.

After resolving on link 10 we obtain the following graph.



Resolving on 4 results in the following graph.

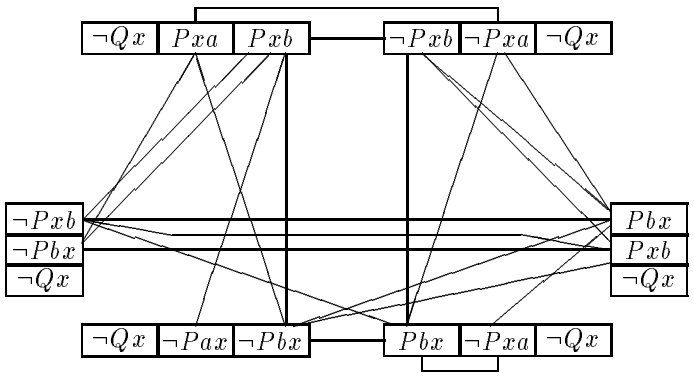


Resolution on 14 leads to $R_{14} : \neg Qa \vee Pba \vee \neg Qb \vee \neg Pbb$, which we do not depict.

Resolving on 13 produces $\neg Qa \vee Pba$, a successive resolution between this clause and $\neg Pba \vee \neg Qa$ leads to a resolvent $\neg Qa$, which subsumes all clauses containing $\neg Qa$. This includes all clauses, which are not depicted.

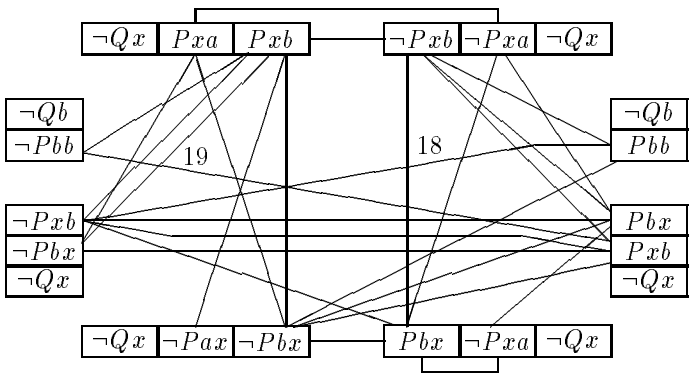
The clause $Qa \vee Qb$ can be resolved with $\neg Qa$ resulting in Qb, which in turn subsumes $Qa \vee Qb$.

The clauses resulting from links 12, 15, 16, and 17 are tautologies. Because the tautology link condition is not fulfilled[(v)], we can only delete the resulting clauses if we do not erase the links from the graph.



In this figure we have all remaining clauses depicted, we only have omitted the clause Qb.

In the next step we infer two factors for the left and right clauses of the graph. For the inheritance in the merging case, mono merging is considered, that is, only the links of one literal are inherited for factors.



[(v)] The clause graph calculus imposes conditions on the existence of special links when deleting tautologies or subsumable clauses (see for example [Bib82]).

Now we have no link between the P-literals of the factors, and hence we shall never succeed to infer the empty clause. The remaining links between maximal literals, that is 18 and 19, lead to dead ends. q.e.d.

The conclusion from this theorem is that the clause graph procedure is inadequate as a basis for superposition calculi, because the restrictions of the two calculi are incompatible. To retain R-link inheritance to be used instead of reconstruction in a superposition calculus we would inhibit the operation links and not inherit the inhibition. Then the clause graph procedure just simulates clause set resolution[vi].

The next theorem connects the clause graph procedure with paramodulation. In this case the properties of clause graph resolution can be retained (but this is relatively worthless when we consider the result of theorem 3.6).

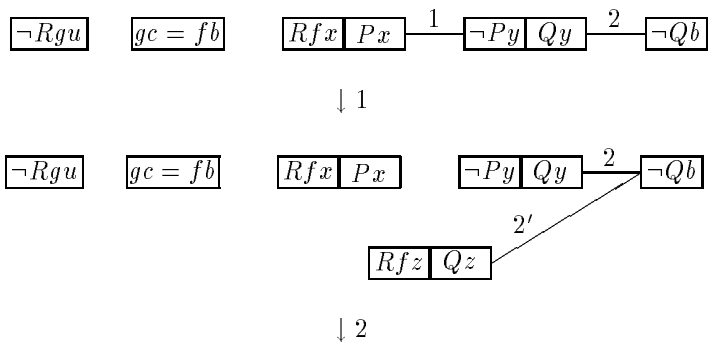
Theorem 3.7 (Paramodulated Clause Graphs)
The clause graph procedure together with paramodulation and the following link inheritance, inference, and construction rules is as complete as the clause graph procedure for resolution.

- *Delete operation link.*
- *Reconstruct all P-links to paramodulants and resolvents.*
- *Construct all R-links to the paramodulated literal.*
- *Inherit all other R-links.*

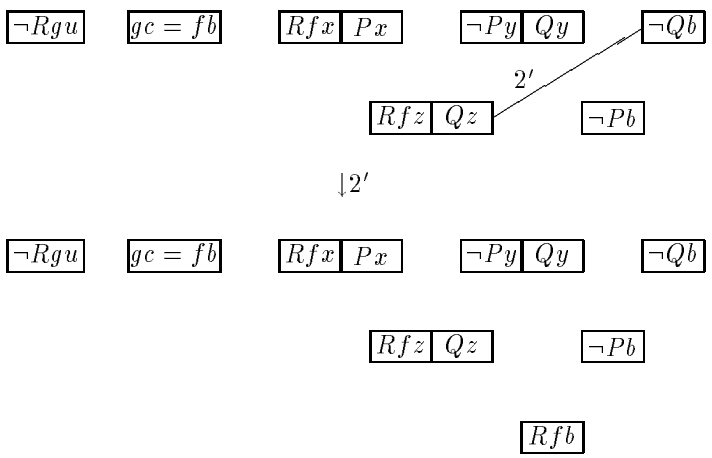
Proof: Each paramodulation step can be simulated by resolution steps with the clauses of definition 1.1 (E-clauses) and the literals involved in the paramodulation step. All paramodulation steps in the graph except those that are already performed are always possible because paramodulation links are not inherited but reconstructed. Hence the procedure behaves as if all E-clauses were always fully connected. Imagine we initially started the procedure with the E-clauses, then we would have fewer links than in the actually described version. Hence it is at least as complete as the one started with E-clauses without paramodulation. q.e.d.

Example 3.8 depicts the typical example where some resolution step is not possible because an operated link (1) is not inherited. But it does not matter that after paramodulation on clause $Rfx \vee Px$ or $Rfz \vee Qz$ no empty clause can be derived because the paramodulation step can also be done on the unit clause Rfb[vii].

Example 3.8 (Link Inheritance)



[vi] The one originally defined by Robinson.
[vii] When we construct a mirrored clause graph to the left hand side of the equation we have a situation in which an application of $gc = fb$ to $\neg Rgu$ in the other direction leads to a symmetric effect.



This example shows that no general purity deletion is possible if equations are present. Clauses can just be pure if the predicate of the linkless literal occurs with only positive or negative signs.

3.3 Reduction

For our purpose we are not interested in a confluent and terminating set of conditional rewrite rules but only in a refutation proof for certain theorems. Hence we do not need a reduction procedure producing unique normal forms independent of the invested resources. We use a reduction rule which has much in common with H. Zhang's contextual rewriting, but retains the reduction facilities of the clause graph procedure. In the sequel we relate contextual rewriting and the reduction rules of the MKRP-system.

The clause graph procedure defines a set of reduction rules as described in [Eis88, Prä85, EOP89]: deletion of pure clauses, tautologies, and subsumed clauses, removal of redundant literals (replacement resolution), removal of links that produce tautologies, subsumed, or pure clauses. Of course purity deletion[viii] is not allowed together with equations in general. First we shall motivate that parts of replacement resolution and the subsumption rule are special cases of contextual rewriting (see definition 2.27). Then we shall show that the other rules are compatible with the superposition calculus and define a new reduction operation.

Subsumption for the clause graph procedure is defined as usual. Example 3.9 shows the correspondence to H. Zhang's contextual rewriting.

Example 3.9 (Subsumption and Contextual Rewriting)

$C = Qx \vee Rx$ subsumes $D = Qa \vee Ra \vee Qb$.

If we select an ordering $Q > R$ then Qx is the maximal literal in the clause C and we can construct a conditional rewrite rule $Qx \rightarrow True \vee Rx$. In this case the condition is $\neg Rx$, the context is $\{\neg Ra, \neg Qb\}$.

A contextual rewriting step with the instance $R = Qa \rightarrow True \vee Ra$ of the rule $Qx \rightarrow True \vee Rx$ leads to the true clause $D' = True \vee Ra \vee Qb$ because the condition literal Ra of R can be reduced to $True$ by the context literal $\neg Ra$ of D.

[viii] The deletion of literals without R-links.

Replacement resolution is a hyperstep in the clause graph procedure combining resolution and a successive subsumption of one parent of the resolvent by the resolvent. This step can be abbreviated by deleting literals in the parent and not explicitly generating the resolvent.

Definition 3.10 (Replacement Resolution)

$$\begin{array}{l} C_1,\ \ldots,\ C_n \\ L_1,\ \ldots,\ L_m,\ K_1,\ \ldots,\ K_k \\ \downarrow\!\!\!

Definition 3.13 (Replacement Paramodulation)

$$\begin{array}{l} C_1, \ldots, C_n \\ L_1, \ldots, L_m, K_1, \ldots, K_k \\ \downarrow\!\!\!

3.4 Summary

We presented two results in this chapter: first we succeeded to combine the difference reduction and the rewrite (superposition) approach in one system using a standard technique, namely narrowing.

The second contribution of this chapter is that the most powerful parts of the MKRP-system can be retained when integrating the superposition approach as a paramodulation strategy.

We developed a clause graph strategy for superposition. Although incomplete, the clause graph strategy induces a behaviour of the MKRP-system as before, when it is only started with clauses without the equality predicate. It behaves like the Knuth-Bendix completion procedure if applied to unit equations.

If controlled by a conventional superposition strategy the theorem prover is complete but has a different behaviour than the original MKRP-system, although the reduction facilities are still applicable.

Chapter 4

Heuristic Control

A calculus only provides inference rules to be applied indeterministically. For an effective application of such a rule system we need special control strategies, that is, rules to instruct the system which step should be performed next. Hence the question arises of how the known heuristics of the field can be used to control our rule system.

After focusing on a global search behaviour concerning graph structures, that is, the E-resolution part of the calculus, we elaborate in the following parts of this chapter upon heuristics to select inference steps and orderings for rewriting.

4.1 Narrowing Control

It is a good tradition in AI to use human problem solving as a guide line for the construction of machine oriented problem solving techniques. For example, A. Bundy developed methods to use human knowledge in equational theorem proving [Bun74, Bun83], although at the lowest level.

In some cases it is relatively simple for a human being to detect the structure of the E-refutation-graph (see section 4.1.1) for a given problem. Two clause graphs for such problems, which can be instantiated to refutation graphs, are depicted in figure 4.1. It is possible to simulate such a human way of solving the problem on a machine by extending the usual difference reduction heuristics (see section 2.4).

At the beginning we shall demonstrate this principle with examples taken from the theory of commutative, zero divisor free rings. The ring axioms are given as usual and the property that the ring has no zero divisors is expressed by a conditional equation $\forall x, y : x \cdot y = 0 \Rightarrow x = 0 \vee y = 0$, which can be transformed into a clause $x \cdot y \neq 0 \vee x = 0 \vee y = 0$.

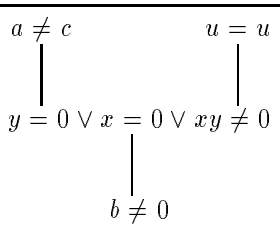
Next we give a proof of a human mathematician at a high level, then we show that it cannot be used to construct an adequate paramodulation proof, and finally we shall illustrate that heuristics can be used to control narrowing in the direction of finding a refutation graph which reflects the structure of the human proof.

Example 4.1 (Human Proof for Cancellation Law)

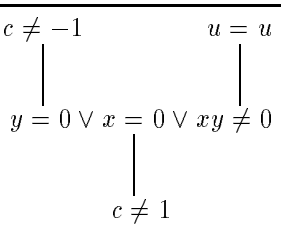
Let a) $(R, +, \cdot, 0, 1)$ *be a commutative ring with 1 and*
b) $\forall x, y : x \cdot y = 0 \Rightarrow x = 0 \vee y = 0$ *(zero divisor free)*
then $\forall x, y, z : x \cdot z = y \cdot z \wedge z \neq 0 \Rightarrow x = y$ *(cancellation)*

Proof: Let $x, y, z \in R$ *and* $x \cdot z = y \cdot z \wedge z \neq 0$
$\Rightarrow x \cdot z - y \cdot z = (x - y) \cdot z = 0$
$\Rightarrow x - y = 0$ *because* $z \neq 0$ *and b)*
$\Rightarrow x = y$
q.e.d.

Figure 4.1: Examples E-Clause-Graphs

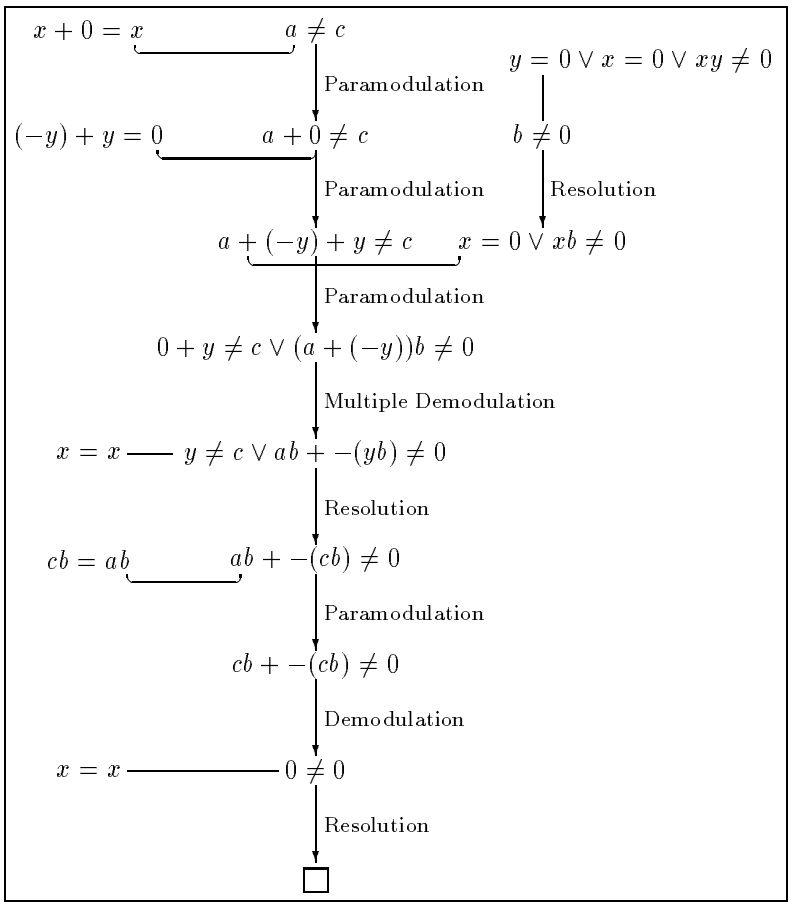


Cancellation in zero divisor free rings (see B.3.1). a, b, and c are Skolem constants stemming from the theorem $x \cdot z = y \cdot z \wedge z \neq 0 \Rightarrow x = y$. The E-refutation graph can be generated by an instantiation with the substitution $\{y \leftarrow c + (-b),\ x \leftarrow a,\ u \leftarrow a(c + (-b))\}$ (see example 3.3).



Square roots in zero divisor free rings (see B.3.2). c is a Skolem constant stemming from the theorem $x^2 = 1 \Rightarrow x = 1 \vee x = -1$. A substitution for the instantiation to an E-refutation graph is $\{y \leftarrow 1 + c,\ x \leftarrow 1 - c,\ u \leftarrow (1 - c)(1 + c)\}$.

Figure 4.2: Paramodulation Proof for Cancellation



In figure 4.2 we depict a paramodulation proof for this example in form of a graph. The proof is hand-made with the following heuristics in mind: use the human proof as orientation, make the proof as linear as possible, and begin the linear chain with a clause in a rather small set-of-support, that is, not the whole theorem but just an essential part of it. The negated and Skolemized theorem consists

of three clauses $cb = ab$, $b \neq 0$, and $a \neq c$ with Skolem constants a, b, and c. The most restricted set of support consists of just the conclusion of the theorem $a \neq c$ and this should be the nucleus of our linear proof. The first action consisting of two paramodulation steps is to expand one side of the inequality by subtracting and adding the same thing, this "something" is intended to become a "c" so that the "something"-literal is $c \neq c$ and can be resolved away. This goal can almost be achieved by applying a rewrite rule to make $a + (-y)$ to 0 but we just have a conditional equation (derived via a resolution step at the right hand side of the picture) and so we introduce a new literal. The next two steps (Multiple Demodulation and Resolution) are done according to our intention of removing the $c \neq c$ literal. What remains to be done is to eliminate the newly introduced literal, which is done straightforwardly by applying a structurally very simple equation (due to its lack of variables) and demodulating until resolution of the empty clause is possible.

This proof seems to be simple but there are essential disadvantages: rewrite rules are used for superposition steps in the reverse direction, that is, unlike narrowing ($x + 0 = x$, $(-y) + y = 0$), paramodulation with variables is used ($x + 0 = x$), associativity is used implicitly in both directions ($a+((-y)+y) = (a+(-y))+y$), and a partially completed set of axioms is used (right identity although it is not given).

Hence the heuristic approach with the human proof as a guide line is not very promising at this level. But this does not mean that heuristics are completely worthless.

With B.3.2 (see right hand side of figure 4.1 and also [Prä90]) we gave a second example of the same class of problems (two square roots), which is essentially more complicated in practice, because both equality literals in the zero divisor clause must be paramodulated into, in order to obtain the empty clause. But the refutation graph has the same structure.

Hence the refutation graphs are relatively simple for such equality reasoning problems and using difference reduction methods at this global level may be valuable. In some sense the heuristic "Produce the shortest clause next!" is also a difference reduction heuristic: the difference to the empty clause is as small as possible with respect to the length of the clauses. In section 4.1.2 we shall adapt V. Digricoli's [Dig81, Dig85], K. Bläsius's [Blä86], V. Lotz' [Lot88], and M. Fuchs' [Fuc90] heuristics to our goal of reducing the difference to an equality refutation graph but first we have to elaborate upon the representation of information and the methods to compute it.

4.1.1 Prerequisites for E-Resolution

In chapter 3 we presented an E-resolution rule based on narrowing. We propose to store the narrowing trees in the literals. For an efficient search of really resolvable nodes of different trees we use indexing trees for every predicate, representing all literals occurring in clauses and narrowing trees with P and $\neg P$.

In order to make the decision about the next narrowing tree node that should be expanded we need data structures to store information about the probability for two nodes to become resolvable. For this purpose PE-links are used.

Definition 4.2 (Potential Equality Resolution Link)
Two literals in different clauses with same predicates and different signs are connected via a **PE-link**.

Our narrowing starts with arbitrary literals and successively searches for the best fitting opposite literal. Therefore it is not advisable to implement PE-links like classical R-links. This would restrict the narrowing to pairs of literals and enhances the amount of multiple computations. We would propose indexing trees for each predicate and sign. Hence PE-links are just virtual links and we need them only to speak about.

Indexing is a special technique to store several terms together in a tree such that algorithms can work simultaneously for parts of the terms. It was presented by R. Overbeek [Ove75] and is based on "tries", a type of trees to store character strings (see for example [AHU83, section 5.3]). An example of an indexing tree is depicted in figure 5.8.

4.1.2 Adaptation of Difference Reduction Heuristics

For the selection of possible equational steps three levels of heuristics can be distinguished.

- At a first level only local heuristic information is exploited: the complexity of critical pairs or literals in the narrowing tree.
- At the second level the difference of two terms that should be made equal is measured. V. Lotz [Lot88] and V. Digricoli [Dig85] mainly define this type of heuristics.
- At the third level we consider several such differences between literals and join them to get information about all resolution possibilities concerning one clause, that is, hyper-E-resolution steps are valued.

Hence a heuristic function to select the next expansion step contains three dependent parts:

$$h(step) = w_1 \cdot h_{local}(step) + w_2 \cdot h_{diff}(step) + w_3 \cdot h_{hyper}(step)$$

step is a node of a narrowing tree, that is, a pair of a literal and a unifier. The three parts of the function h are dependent in the sense that the definition of h_{local} can recursively be used in h_{diff} and in h_{hyper}, as well as the definition of h_{diff} can be used in h_{hyper}.

It is not difficult to imagine several functions to be used for h_{local}, which computes in some sense the size of the narrowing tree node:

- number of variable, function, and constant symbols
- number of all functions
- Knuth-Bendix weight with weights for function and constant symbols
- weighting polynomials (see section 4.2.4)

h_{local}, h_{diff}, and h_{hyper} can be taken just for the literal alone and also for the literal and the unifier together. The main task of h_{local} is to prune the most complex tasks from the search space because the biggest problem of completely uninformed breadth first search is that very complex states must be considered very early in the search process, while with the same effort hundreds of simpler nodes can be expanded.

h_{diff} values the difference of the input node to some other potentially unifiable node. Let $C = L^{t^1,\ldots,t^k}, L_1, \ldots, L_n$ be a clause in a clause set. We denote the first literal with L and this literal is a maximal literal. $t^1, \ldots, t^k$ are the nodes of the narrowing tree attached to L. Each t^i is a pair (L^i, σ^i) with L^i a literal descending from L by successive applications of equations and σ^i the merge of the unifiers of equation applications (see definition 3.4).

For each literal L there exists a set of literals $\{M_1, \ldots, M_m\}$ connected via a PE-link to L. This link expresses the potential unifiability mentioned above. Let D_j be the clause containing M_j. Each literal M_j has narrowing trees with nodes $((M_j^1, \mu_j^1), \ldots (M_j^{n_j}, \mu_j^{n_j}))$. Now we can define a difference measure

$$h_{diff}((L^i, \sigma^i)) := (min_{j=1}^{m})_{k=1}^{n_j}(dist(M_j^k, L^i))$$

The function *dist* is intended to compute the difference between two terms or literals. It can be taken from V. Lotz [Lot88, pages 101-112] or V. Digricoli [Dig85] and is integrated in our heuristic h_{diff} in the introduced form, because firstly V. Lotz and V. Digricoli always consider the possibly applicable equations and secondly they focus on two terms in a path to be compared whereas we want to completely

expand the nodes of the narrowing tree which is not so complicated compared to only applying one equation and to keep book over the used equations. In addition it is more homogeneous to consider only terms and not terms and equations in relation to the terms. In fact it is always better to consider the actual instance instead of the general equation. We think that by this the complexity of the expansion operation is decreased and the operation itself is better informed.

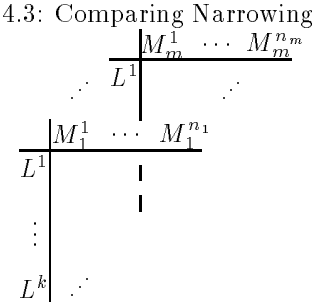
h_{hyper} is intended to compute something like the value of a narrowing step to diminish the difference of a clause to the empty clause. Let C from above be a clause in a clause set with $((L_j^1, \sigma_j^1), \ldots (L_j^{n_j}, \sigma_j^{n_j}))$ for $j = 1, \ldots, n$ the narrowing trees for the literals of C. Then

$$h_{hyper}((L^i, \sigma^i)) := \sum_{j=1}^{n} min_{k=1}^{n_j}(h_{diff}((L_j^k, \sigma_j^k)))$$

The function h_{hyper} values all nodes of narrowing trees in one clause equally, only the combination with h_{diff} and h_{local} characterizes different values. Of course the function can be extended to take into account the interrelations between the n unifiers.

Unfortunately the computation of h_{diff} and h_{hyper} causes problems. In principle we would have to consider several matrices for each PE-link (see figure 4.3) with the dimensions being the actual number of nodes in the narrowing trees in order to get the same heuristic power as V. Lotz and V. Digricoli. Of course these matrices need not always be recomputed, they can be extended when adding new nodes, but the amount of storage and update is enormous, so it seems useful to constrain the usage of h_{diff} and h_{hyper}. Otherwise there would be too much computation.

Figure 4.3: Comparing Narrowing Nodes



One possibility is to only use the difference to some literal in another clause and not to all nodes of the narrowing trees. Then h_{diff} is defined as $min_{j=1}^{m}(dist(M_j, L^i))$. h_{hyper} must be computed only once for each clause when not considering further dependencies.

4.1.3 Restrictions for Narrowing

Unrestricted narrowing makes it necessary to compute indexing trees or related structures for all literals which are not positive equations. Therefore we like to have more semantically motivated restrictions for narrowing steps than those mentioned above to keep the set of trees as small as possible.

In section 3.1.1 we motivated the use of a linear E-resolution strategy based on narrowing because it is a way to "divide and conquer" the whole problem into the subproblems to resolve away single literals. Linear resolution needs a focus clause as nucleus of the inference sequence. Clauses that serve as starting point for E-resolution can be determined in various ways.

- In a semi-automatic proof system the selection can be done by the user.
- If only one non equality literal is present this can naturally be selected as nucleus.

- If only one clause with more than one literal is in the clause set we begin the sequence of inference steps with the maximal literal of this clause.
- It can be advisable to use the theorem or part of it as a focus clause, as it is done in the set-of-support strategy.

The inference system remains complete if these restrictions are only used as a supplement to the original superposition calculus, that is, all superposition steps remain possible. E-resolution is just an abbreviation of several paramodulation steps.
E-resolution steps can be seen as normal superposition steps like resolution itself. The problem with narrowing is how the interleaving of narrowing and superposition steps can be controlled. In the following some control rules are specified.

- If a new literal is created then compute the narrowing tree up to a predefined level. This level is determined by the heuristics in the previous section and some user specified options.
- After the creation of a new clause perform n narrowing steps according to the heuristics.
- After creation of a new unit rewrite rule extend all narrowing trees up to the actual level using this rule.
- If some exception occurs extend all or some selected narrowing trees by one (m) level. Such an exception could be that no unit superposition steps remain to be performed.
- If a supervisor program assigns a high heuristical value to a PE-link extend the trees of the corresponding literals.

All five rules can be combined or used alone. The simplest version of narrowing integration is to only use the first one. Then it is not necessary to update any information, the potentially available narrowing control information is approximated as roughly as possible.

4.2 Completion Control

There are many heuristics in automatic theorem proving and equality reasoning but their respective contribution to the overall search is sometimes dubious. In order to measure their influence, we used some more complex examples, because the loss of accuracy caused by the machine and the graph construction is relatively smaller.
A paramodulation theorem prover based on Knuth-Bendix completion and rewriting usually uses two orderings. The first one is the ordering to constrain the possible steps and to perform reductions. N. Eisinger calls this a restriction strategy [Eis88]. It must be a reduction ordering or the strategy of the theorem prover becomes incomplete. Examples of such restriction strategies for resolution theorem proving are the set-of-support strategy or the basic strategy.
The second one is a selection ordering for the selection of the next step called an ordering strategy by N. Eisinger [Eis88]. This ordering need not necessarily be Noetherian. But it can be more goal directed and more powerful. For example, such a strategy can be chosen to perform goal directed depth first search without considering the termination. The selection strategy can be used to employ a weakened form of a restriction strategy, for example, if this restriction strategy is incomplete. A typical way to implement a selection strategy is to compute some heuristic values for all possible steps.
A possibility to approximate the set-of-support strategy is to multiply the heuristic value of all non set-of-support clauses with a big constant value. By this we obtain a goal directed selection of inference steps even if the set-of-support strategy is incomplete.
Our goal was to examine which problems can be solved with which strategy and hence to provide a basis to automatically choose both, restriction and selection strategy, if the user does not specify them. We shall elaborate the restrictions in the sections 4.2.1 and 4.2.2 and the selection ordering in section 4.2.3.

4.2.1 Choosing the Restriction Strategy

We selected six restriction orderings of paramodulation for our experiments on strategies. H-C and C-G denote our own ones described in section 3.2.2 and abbreviate "Heuristic-Completion" and "Connection-Graph". Both use the inheritance rules of theorem 3.7, but they differ in the selection function for the remaining links. C-G weighs R- and P-links in the same way as all other strategies whereas H-C uses a different resolution specific function for R-links. Hence the H-C strategy includes through its definition a different selection function and is not really comparable to the others. It is only included in the investigation for the sake of completeness of the considered methods.

H-C is intended to retain the behaviour of the MKRP-system if no equations are present. The resolution steps are selected according to the selection function of the old MKRP-system, whereas the paramodulation steps are selected according to a selection function developed for the superposition rule. This causes a disjoint control of resolution and paramodulation for H-C and makes it necessary to correlate the two heuristic values with a factor. Hence this strategy is not well suited for the selected examples, which combine throughout equational and non-equational parts. It remains to be noted that H-C uses a strong set-of-support selection for resolutions.

C-G serves as a test for the link inheritance for ordered paramodulation and resolution. Although incomplete, in practice such a proceeding can be useful. The clause graph procedure behaves often good-naturedly even if it is controlled with an incomplete restriction strategy.

Z-K stands for "Zhang-Kapur" (see definition 2.22, strict clausal superposition), B-G for "Bachmair-Ganzinger" (see definitions 2.23 and 2.24, strict and merging superposition).

S-L abbreviates "Snyder-Lynch" (see definition 2.26, basic paramodulation). It provides an implementation of basic paramodulation (the variant without dagger).

D stands for "Dershowitz" (see definition 2.25, unit superposition and narrowing). We implemented a version of his unit strategy that is incomplete in the case where non-Horn clauses are in the input set. Another possibility would be to check whether the input clauses are Horn clauses and in case they are not Horn to select a different complete restriction strategy.

As examples we choose the conditional equality examples of F. Pelletier and a famous non equality one: Schubert's steamroller problem (published for example as problem 47 of F. Pelletier). Automatically generated proofs for all equality examples can be found in appendix B. The most examples in the appendix are computed using the B-G strategy.

For all examples computed here we could realize that a slight change of the selection strategy (heuristic) and of the ordering (see the lines 51a and b of table 4.1) has a stronger influence on the search behaviour and hence on the run times than choosing a different restriction strategy.

Unless otherwise stated we used the automatically generated ordering.

$\bowtie$ indicates a collapse of the graph for an incomplete strategy, which is often the case for the D-strategy, because the selected examples are non-Horn problems. ∞ means that the system was not able to find a proof within a reasonable time.

Z-K/B-G One advantage of B-G over Z-K is its completeness when deleting tautologies. Additionally B-G considers the substitution for the restriction: the terms and literals have to be maximal in the instances of the clauses. Example 47 shows that the consideration of the substitution can be necessary for finding a proof.

C-G Surprisingly there are really two examples (51a, 55) which behave best with this strategy, that is, sometimes it is good to use inheritance for the resolution case.

Table 4.1: Different Strategies (Run Time in Seconds)

Example		H-C	C-G	Z-K	B-G	S-L	D
47	a	∞	133	∞	127	127	$\bowtie$
	b	45	54	34	34	34	34
48(B.2.1)		6	6	6	6	6	6
49(B.2.2)		12	12	12	13	13	14
51(B.2.3)	a	∞	23	43	33	33	∞
	b	∞	28	48	48	41	∞
53(B.2.5)		∞	16534	2181	2137	2732	$\bowtie$
54(B.2.6)	a	∞	25	29	31	31	$\bowtie$
	b	74	25	29	31	31	$\bowtie$
55(B.2.7)		23	17	22	22	21	$\bowtie$
56(B.2.8)		9	9	9	9	9	$\bowtie$
73(B.2.14)		∞	∞	∞	5929	5898	$\bowtie$

47 The ordering for the steamroller was *Grain* > *Snake* > *Caterpillar* > *Bird* > *Fox* > *Wolf* > *Plant* > *Animal* > *Smaller* > *Eats*. With other orderings the system did not succeed to find a proof in any strategy: we tried *Smaller* > *Eats* > *Plant* > *Grain* > *Snake* > *Caterpillar* > *Bird* > *Fox* > *Animal* > *Wolf* (automatically generated) and *Plant* > *Animal* > *Grain* > *Snake* > *Caterpillar* > *Bird* > *Fox* > *Wolf* > *Smaller* > *Eats*.

This is not surprising, the successful ordering is constructed using the sort hierarchy of the example. In this case the ordering is a different way to express such a sort hierarchy.

47a Without terminator.

47a/H-C Example 47 could not be proved with H-C because the strong set-of-support strategy is not well suited for such examples.

47b With terminator [AO83]. H-C and C-G need 26 resolution steps before the proof is found by the terminator whereas for the other strategies the proof is found immediately after the first step. The set-of-support strategy in H-C and B-G prevents inference of units useful for the terminator but the other strategies need more time to compute possible results.

49 B-G and S-L need more steps. D generates a completely different proof.

51a Ordering $P > f_2 > f_3 > f_1$.

51b Ordering $f_2 > f_3 > f_1 > P$.

51/H-C Because of the wrong relation between paramodulation and resolution steps the problem could not be solved in a reasonable time.

53 Ordering $f_5 > f_2 > f_3 > f_4 > f_1 > P$ and $f_7 > f_9 > f_{10} > f_6 > f_8 > P$.

54 Could only be solved with terminator.

54a Ordering $In > f_4 > f_3 > f_2 > f_1$ (Skolem functions as normal functions).

54a/H-C For resolvents the old MKRP-strategy does not consider the term depth for the weight.

54b Ordering $f_4 > f_3 > f_2 > f_1 > In$ (Skolem functions greater than all others).

55 Longer proofs for B-G and S-L.

The first question to be answered from these results is: did it pay to integrate the various superposition strategies into the MKRP-system? For all examples in table 4.1 the H-C strategy which is defined to retain the behaviour of the old MKRP-system is worse than Z-K and B-G. This is not surprising because the examples in the table are just of the type that can not be handled by the old system. Example 47a shows that a restriction strategy can even enhance the power of the system for pure resolution examples, but only when we choose a very dedicated ordering.

In 47b which behaves better for all strategies than 47a an MKRP-feature, the terminator, is used. This is also the case for example 53, where the splitting and the finite domain options are set. That is, the integrated features are useful throughout all strategies.

The second question is, when to use which strategy. For all examples the B-G strategy behaves best concerning computation time. Hence it pays to consider the instantiation caused by the substitution, when deciding the next step. The expected effect that this strategy often produces longer proofs does not happen as shown in table 4.2. This effect is a property of many restriction strategies in automated theorem proving (see also section 3.1).

Table 4.2: Different Strategies (Proof Length in Steps Including Rewrites)

Example		H-C	C-G	Z-K	B-G	S-L	D
47	a	∞	40	∞	40	40	$\bowtie$
	b	52	52	45	45	45	45
48(B.2.1)		9	9	9	9	9	9
49(B.2.2)		15	16	15	15	15	16
51(B.2.3)	a	∞	17	17	17	17	∞
	b	∞	17	17	17	17	∞
53(B.2.5)		∞	597	564	564	580	$\bowtie$
54(B.2.6)	a	∞	22	22	22	22	$\bowtie$
	b	23	23	24	24	24	$\bowtie$
55(B.2.7)		17	15	15	15	15	$\bowtie$
56(B.2.8)		15	15	15	15	15	$\bowtie$
73(B.2.14)		∞	∞	∞	195	195	$\bowtie$

4.2.2 Reduction Orderings

For the MKRP-system it is possible to select a Knuth-Bendix ordering, a polynomial ordering, or a lexicographic recursive path ordering (LRPO). If the user does not specify the ordering for an operator the system automatically extends the specified ordering or completely generates one.

We mainly experimented with LRPOs. An exception is the case of theory unification, where we used polynomial orderings because the restriction on the ordering caused by the theory can be more simply modelled in the polynomials.

Some of the results in table 4.1 show that the choice of the ordering can be essential for finding a proof. Example 47 can only be solved with an ordering reflecting the sort hierarchy inherent in the example. Examples 51a and b as well as 54a and b also show that the ordering has influence on the run time and the proof length.

In this section we shall analyse the choice of the operator ordering for example 53. We shall show that the influence of the reduction ordering is essential for the run time even if all selected orderings are total because the example contains only ground terms. We selected this example because it is important to choose good orderings especially for Skolem functions[(i)], which cannot be a priori chosen automatically.

[(i)] Skolem functions are automatically generated by the theorem prover for all existentially bound variables.

The automatic generation for the operator ordering of the LRPO is according to the following rules. For the description of the implemented ordering selection we use the term "function" for functions as well as for constants. A symbol f is selected to be greater than g if

1. both are functions and the sort of f is greater than that of g,
2. f is a Skolem function and g not,
3. f is a predicate and g is a non-Skolem function, or
4. both are predicates or functions, and the arity of f is greater than that of g and g is not a Skolem function.

If both are equal relative to this specification we order them arbitrarily.
The reasons for the rules are as follows:

1. The MKRP-system is an order-sorted theorem prover [Wal84]. It is known that the ordering for the reduction relation must be compatible with the sort hierarchy [Sch88]. An example is the refutable clause set $\{R(a_T), \neg R(x_S), a_T = b_S\}$. The sorts are written as indices. S is a subsort of T. Therefore x_S and a_T are not unifiable. a_T must first be replaced by b_T which would not be possible if we select an operator ordering $b_S > a_T$ opposite to the sort hierarchy.
2. Skolem functions are auxiliary functions to be deleted as soon as possible, hence we choose them greater than others.

   Making Skolem functions as big as possible is obviously not always a good idea as the following example shows:

   Let $S(y,x) \vee A(f(x,y),y)$ be a clause. If we select an ordering first working on the S literal because we think the A literal will be reduced away after a fitting resolution on S and select an ordering $S > A$ then we have to choose $S > f > A$ to enforce $L_1 > L_2$. If we select automatically $f > S > A$ then we have $L_2 > L_1$ which is not the idea of $S > A$.
3. We select predicates greater than functions because it is an advantage for clauses like $P(x,y) \vee f(x,y) = x$ if the non-equational literal is maximal. This reduces the number of possible operations because resolution is less branching than paramodulation (see also section 1.2).
4. If we have for example a clause $P(x,y) \vee Q(x) \vee R(y)$ the only possibility to have a single maximal literal in the clause is an operator ordering with $P > Q$ and $P > R$. With $Q > P > R$ as well as with $R > P > Q$ we obtain two incomparable maximal literals. All other orderings even lead to three incomparable maximal literals. The variable distribution as in the clause above is typical for the relation of predicates and also of functions with different arity in the same clause. Therefore we select a predicate P greater than Q if its arity is greater, and correspondingly proceed for functions.

Next we tried to refine the rules for the automatic generation of the operator ordering. We computed example 53 with various permutations of the ordering for the five Skolem functions. Table 4.3 shows that three behavioural variants can be distinguished.
Table 4.3 shows that it is essential to have $f_2 > f_1$. If additionally $f_1 > f_5 > f_2 > f_3 > f_4$ we have an even worse behaviour. We try to correlate the condition $f_2 > f_1$ with the occurrences of the functions in the clause which we depict in table 4.4.
f_2 only occurs together with f_1 whereas f_1 also occurs with another function (f_3) in the same clause. f_2 must be greater than f_1 to replace the f_2 in this clause by f_1 (see section B.2.5, `D228` is the conditional rewrite rule). This examination induces a refinement of the ordering rules: select the function or predicate as the biggest one which has the least dependencies with other functions or predicates.
It remains to examine other examples, to implement and verify this refinement of ordering selection, and to integrate other approaches for the automatic generation of orderings. It is also necessary to transmit the rules to polynomial orderings, for which it is essentially more complicated to define a fitting ordering.

Table 4.3: Statistics For Different Ordering of Skolem Functions

Ordering	Proof Length	Time
$f_5 > f_2 > f_3 > f_4 > f_1 > c_3$	564	2137
$f_2 > f_5 > f_3 > f_4 > f_1 > c_3$	564	2166
$f_2 > f_3 > f_5 > f_4 > f_1 > c_3$	564	2180
$f_2 > f_3 > f_4 > f_5 > f_1 > c_3$	564	2197
$f_2 > f_3 > f_4 > f_1 > f_5 > c_3$	564	2157
$f_3 > f_2 > f_4 > f_1 > f_5 > c_3$	564	2173
$f_3 > f_4 > f_2 > f_1 > f_5 > c_3$	564	2176
$f_3 > f_4 > f_1 > f_2 > f_5 > c_3$	709	4832
$f_3 > f_4 > f_1 > f_5 > f_2 > c_3$	709	4876
$f_4 > f_3 > f_1 > f_5 > f_2 > c_3$	709	4854
$f_1 > f_5 > f_2 > f_3 > f_4 > c_3$	1002	10661
$f_5 > f_2 > f_1 > f_4 > f_3 > c_3$	564	2171

Table 4.4: Occurrences of Skolem Functions

Function	Arity	Occurrences in Clauses
$f_1 \& f_8$	1	8
$f_2 \& f_9$	2	4
$f_3 \& f_{10}$	1	2
$f_4 \& f_6$	1	2
$f_5 \& f_7$	2	4
$c_3 \& c_4$	0	4

- $2 \times f_1$ and f_2 in same literal
- $2 \times f_3$ and $2f_1$ in same clause
- $2 \times 2f_1$ and $2f_2$ in same clause
- $2 \times f_4$ and c_3 in same clause
- $2 \times 2f_5$ and c_3 in same clause

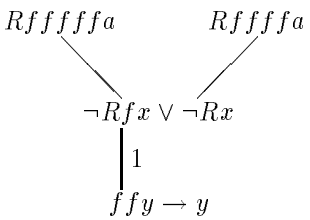
4.2.3 Critical Pair Selection

In principle it is sufficient to have a reduction ordering around and to abuse it also as a selection ordering, but this is a restriction in usefulness and efficiency.

We begin this section with some differences of selection strategies in Knuth-Bendix completion and arbitrary theorem proving. It is known that a selection function counting the number of symbols in the critical pairs gives a complete strategy for completion when keeping the rules interreduced[ii]. Such a selection is called "smallest component strategy" by G. Huet [Hue80]. But this is not necessarily the case for general theorem proving as shown in figure 4.4. The reduction relation must be enhanced such that it contains at least subsumption.

Figure 4.4: Example for Incomplete Counting Strategy

In this example the clause $\neg Rfx \vee \neg Rx$ remains the same after each paramodulation step on the P-link labeled with 1 and the correspondingly constructed link is always selected anew.



For a set of selection strategies see J. Denzinger in the description of his experts [Den91]. He also gives a detailed discussion and measurements for examples in the case of unfailing completion with goal, that is, the restriction on the input is: the axioms are all unit equations and the theorem is one disequation.

[ii] That is, each rule is completely reduced by all other rules in the actual rule set.

Hence we concentrate on the conditional case with our own measurements. The MKRP-system allows to specify polynomial weighting functions for all predicate, function, and constant symbols. We do not vary this parameter, we just use the default weighting which counts the symbols. In the sequel the result of the polynomials is denoted by "default-weight". Again we use the examples of section 4.2.1.

First we explore the influence of a set-of-support weighting on the run times[iii]. MKRP supports the weighting of set-of-support clauses for examples by specifying for example `(* (if support 1 2) ...)` in the weighting function[iv]. In table 4.5 we used the following weighting functions.

1. `(*` default-weight `(if support 1 10))`
2. `(*` default-weight `(if support 1 3))`
3. `(*` default-weight `(if support 1 1.5))`
4. default-weight
5. `(*` default-weight `(if support 1.5 1))`
6. `(*` default-weight `(if support 3 1))`
7. `(*` default-weight `(if support 10 1))`

The parameter setting 1 corresponds to a strong set-of-support selection, the setting 7 simulates a forward reasoning strategy. The variation from 1 to 7 causes the transition from backward to forward reasoning.

Table 4.5: Set-of-Support Variation (Run Time in Seconds)

Example	1	2	3	4	5	6	7
47a	∞	∞	∞	254	230	134	134
48(B.2.1)	7	7	7	7	7	7	7
49(B.2.2)	13	13	13	13	13	13	13
51(B.2.3)a	27	27	27	27	27	27	27
53(B.2.5)	2180	2180	2180	2180	2180	2180	2180
54(B.2.6)a	23	23	23	23	23	23	23
55(B.2.7)	22	22	22	22	22	35	35
56(B.2.8)	9	9	9	9	9	9	9

The reason for the times of example 47 is that the ordering forces the system to first eliminate the sort literals and then to do the main proof. This can only be done when first resolving between axioms, otherwise the sorts in the axioms cannot be removed. For the other examples the set-of-support plays no or no essential role. We also tested the set-of-support weighting for pure equational examples, where it is typical that equations are not applicable to the theorem in the initial clause set.

The next parameter we vary is the weighting of the depth of the search space.

1. `(+` default-weight `(* depth 10))`
2. `(+` default-weight `(* depth 3))`
3. `(+` default-weight `depth)`
4. default-weight

[iii] We did not use any special features and always selected the B-G strategy. Unless specified otherwise we used the same ordering as in section 4.2.1.

[iv] The critical pair clause with the lowest value is selected.

5. `(-` default-weight `depth)`
6. `(-` default-weight `(* depth 3))`
7. `(-` default-weight `(* depth 10))`

The parameter setting 1 induces a behaviour close to breadth first search, the setting 7 approximates depth first search, that is, the variation from 1 to 7 causes the transition from breadth to depth first search.

Table 4.6: Depth Variation (Run Time in Seconds)

Example	1	2	3	4	5	6	7
47	317	254	254	254	254	254	134
48(B.2.1)	7	7	7	7	7	7	7
49(B.2.2)	13	13	13	13	13	13	13
51(B.2.3)a	93	63	27	27	23	23	23
53(B.2.5)	3230	4056	2600	2282	4687	6646	6687
54(B.2.6)a	64	38	32	23	23	24	∞
55(B.2.7)	22	22	22	22	22	22	27
56(B.2.8)	9	9	9	9	9	9	9

From the results in table 4.6 we can conclude that it is a good heuristic to ignore the depth of a proof step completely. Only for examples 47 and 51 we realized a slight improvement when we prefer deeper steps, but this seems to be an accident. Setting 7 for example 54 is the only one with the depth first effect of getting lost in the depths of the search space.

The different results in table 4.6 show that an extension of a multi agent approach as proposed by J. Denzinger (see [Den91]) to restriction strategies is also very promising for the conditional case.

4.2.4 Critical Pair Reduction

A global design decision is when to reduce potential results of superposition steps.

Essentially we have three possibilities how critical pairs can be administrated with regard to reduction. The question is, how much information should be used to make the decisions of the previous section.

1. As a maximal solution we can keep the critical pairs reduced all the time exactly as it is done for rewrite rules.
2. We can reduce them after creation and after selection. That is the most natural way to proceed. Whenever you create an object you will reduce it, in the first case a reduced critical pair is created, in the second one a reduced rewrite rule.
3. As a minimal solution we can just reduce them after selection. That is necessary to get an interreduced set of rules but the reduction facility is absolutely not exploited for the selection of the next rule.

In table 4.7 we give the execution times and the numbers of performed steps for three examples. The table shows that it pays to perform reductions whenever it is possible. This is valid for all examples with a high number of successful reductions. For systems with many rules which are seldomly applicable the second strategy should be selected, this selection can be controlled automatically by storing the numbers of successful and successless applications of rules and invoking the first strategy if the statistic limits are exceeded.

Table 4.7: Statistics, Reduction of Critical Pairs

Example	Times in Seconds			Proof Steps		
	1	2	3	1	2	3
B.3.4 (Group)	16.57	17.06	30.60	13	14	39
B.3.5 (Groupoid)	81.84	100.75	198.43	31	42	145
B.3.6 (Z22)	3196.05	4701.91	∞	91	121	∞

4.3 Summary

First the heuristics developed for difference reduction are applicable in the combination of superposition and narrowing. They can even be extended to cope with something like a refutation graph structure.

The second result concerning heuristics is that those developed for resolution are also compatible with the superposition approach. Resolution strategies that are incomplete in the presence of equations can be simulated by an approximation with heuristics.

Naturally the heuristics for rewriting systems are included in a proof system based on superposition.

Chapter 5

Compilation: Towards an Equality Reasoning Machine

How can a superposition calculus be implemented efficiently using standard techniques of automated theorem proving like indexing, structure sharing, and compilation, without loosing facilities like heuristic search, hashing, and weights? We shall show how these techniques can be retained by systematically constructing an abstract machine for rewrite operations.

The presented actual prototype realization on a SUN Sparc architecture (see appendix A) did not come up to our expectations based on experiments with compilation in Lisp, but shows that a compilation approach is in principle compatible with all tools not relying on special data structures as it is the case for various theory unification algorithms.

That the results are not convincing was the reason to not incorporate the extensions of the sections 5.4 and 5.5 into the SUN implementation.

How to compile and apply canonical rewrite systems is known from rewrite rule compilation [Kap87] and Prolog techniques. Especially in the **R**ewrite **R**ule **M**achine (RRM) project an advanced efficient and parallel reduction machine is under development [AGM90, Gog90]. It bases on an array architecture consisting of many small processing cells with local storage. But such a reduction machine always requires a fully specified canonical rewrite system such that it can be compiled on a special machine. Another disadvantage is that it is not dedicated to the standard von Neumann architecture. In the RRM-project a hardware realization [ALM90] is favoured. Our goal is not such a functional language compiler but instead we are interested in fast equality reasoning.

The second compilation approach in the context of automated reasoning is the **W**arren **A**bstract **M**achine (WAM) for Prolog [War77]. This method is not adequate for equality reasoning because every serious equality problem needs dynamical changes of the structure, thus violating one of the basic assumptions of compiled Prolog.

Prolog is a programming language but also a special proof procedure: the three disadvantages of this special purpose strategy, namely, its restricted depth first and ordering strategy, the lack of an occur-check in the unification procedure, and the confinement to Horn clauses, can be avoided as suggested inter alia by M. Stickel [Sti86]. This extended Prolog is a complete first order theorem prover. However the most interesting feature of Prolog is a programming language feature, namely the possibility of compilation. Incorporating equality into Prolog means in the programming language context to introduce functional programming into logic programs.

There seem to be five distinguishable approaches to incorporate equality into Prolog, namely: J. Jaffar, J.-L. Lassez, and M. Maher [JLM84], J. Goguen and J. Meseguer [GM85], SLDE-resolution of J. Gallier and S. Raatz [GR89], N. Dershowitz and D. Plaisted [DP85], and L. Fribourg [Fri84b, Fri85b]. The

first three are E-unification based (see section 2.3) whereas the other two build upon superposition (see section 2.6).

The E-unification based approaches are easier to incorporate into existing non-equational Prolog-systems but they are less specialized to equational reasoning and hence less effective in proving theorems than the others (see sections 2.4.2 and 2.6).

P. Baumgartner developed a technique to transform a canonical rewrite system directly into Prolog clauses [Bau90]. He uses a restriction of the equality axioms (see definition 1.1), which is a simple one but may be efficient for logic programs if the role of equations is not dominating.

The main example of J. Gallier and S. Raatz in [GR89] shows the inefficiency of E-unification based systems when compared to rewriting.

Example 5.1 (Gallier-Raatz) *The first column of table 5.1 gives the set of Prolog clauses to be refuted, where the usual Prolog syntax :- is used. We begin by reducing the first clause*[(i)] *with* $b \to a$ *and deletion of the false literal and arrive at the second column with a new rewrite rule* $f^3a \to a$*. With it the fifth clause can be reduced resulting in* $Q(a)$*. Then clause 4 is reduced by* Qa *producing a new rule:* $f^2a \to a$*. This rule reduces* $f^3a \to a$ *to* $fa \to a$ *depicted in the fourth column. After several reductions we obtain a refutation only by simplifying the set of clauses and one final resolution step between* Ra *and* $\neg Ra$*.*

Table 5.1: Rewriting the Gallier-Raatz Example

$f^3a = a$:- $fa = fb$	$f^3a = a$	$f^3a = a$	$fa = a$
$a = b$	$a = b$	$a = b$	$a = b$
Pa	Pa	Pa	Pa
$f^5a = a$:- Qa	$f^5a = a$:- Qa	$f^2a = a$	$fa = a$
Qa :- $f^3a = a$	Qa :- $f^3a = a$	Qa	Qa
Ra :- $fa = a, Pfa$	Ra :- $fa = a, Pfa$	Ra :- $fa = a, Pfa$	Ra
:- Rfa	:- Rfa	:- Rfa	:- Ra

The advantage of the E-unification based systems is that they can be handled also by Prolog compilers and hence the non-equational part is very efficient.

The best known method to compile logic programs is that based on the Warren Abstract Machine (WAM) of D. H. D. Warren [War77, War83, GLLO84]. The main idea is to compile each head literal of a clause into a self unifying piece of code representing a specialized unification algorithm instead of using a general one. In addition a set of stacks are needed to handle backtracking and bindings.

Two operational models of functional languages can be incorporated into an abstract machine like the WAM. Environment based graph reduction machines were developed for example by J. Fairbairn and S. Wray [FW87] or S. Peyton-Jones and J. Salkild [PS89]. A machine based on lazy narrowing is developed by M. Chakravarty and H. Lock [CL91].

We do not think that Prolog techniques extended in the E-unification style are very useful in general theorem proving with equality due to similar reasons as general E-unification. The disadvantage of the superposition style is that it is not really Prolog and that it is therefore not better than the normal superposition calculi.

M. Stickel's PTTP[(ii)] idea [Sti86] (iterative deepening) is not as promising as for pure resolution systems because of the high branching rate of equation applications.

[(i)] An orientation in the opposite direction makes no essential difference.

[(ii)] **P**rolog **T**echnology **T**heorem **P**rover

There are various approaches to integrate functional and logic programming on the basis of the WAM by combining the two techniques as for example the attempt of P. Kreuger [Kre89] or M. Chakravarty and H. Loch [CL91].

The essential point used in our work is the idea of compilation. In this chapter we shall focus on the main extension of our system: newly derived rules. Hence we improve the compilation methods to an algorithm which automatically derives new rewrite rules and which is in some sense an extension of both, Prolog-WAM and compiled rewrite rules. The power of the WAM is extended to work with arbitrary clauses and rewrite rule applications to perform superposition. Our abstract machine is not intended to be as complete as the WAM is, but only to give machine operations for the special inferences in our context, thus our approach is dedicated to a standard machine architecture and gives a procedure to semi-automatically generate code to handle different standard architecture machine code.

The main idea is to perform superposition and unification on the compiled program. The actual development of the algorithms is done in two steps:

1. Lisp programs are given to perform unification of interpreted Lisp functions in λ-notation.
2. Equivalent C programs are specified to perform the operations on pieces of assembler code and to generate new pieces of code for the newly derived rules.

The concrete procedure is performed for a Sparc-machine realization in appendix A. The method presented here is general in nature and hence can be applied to any kind of inference if the application of rules via matching is the dominating operation.

Beginning with the pure Knuth-Bendix algorithm we first give a sequence of transformations which finally generate an efficient implementation of the matching algorithm, which is the basis of the abstract machine. This transformation shows the algorithm to be as correct as the original calculus. The proceeding is general enough to work with the Bachmair-Ganzinger calculus or any other calculus based on superposition. The system is actually only implemented for the pure Knuth-Bendix procedure.

It is clear that one needs a method to specify the correspondence between the abstract reasoning machine and the special assembler language, because in contrast to usual compilers a method to newly construct code (terms) must be given. This can be done relatively easily because we defined a narrow interface and used just a few assembler instructions.

5.1 A Compilation Example

We begin this chapter with an example. The left hand side of the rule $-u + u \rightarrow 0$ can be overlapped into the term $x + y$ of the rule $(x + y) + z \rightarrow x + (y + z)$ with a unifier $\{x \leftarrow -u, y \leftarrow u\}$. For the following discussion we give names to all occurring subterms as listed in table 5.2.

Table 5.2: Term Names

Term	Name	Term	Name	Term	Name
$(x + y) + z$	l	$x + (y + z)$	r	$-u + u$	l'
$x + y$	l_1	x in r	r_1	$-u$	l'_1
x in l	l_{11}	$y + z$	r_2	u in l'_1	l'_{11}
y in l	l_{12}	y in r	r_{21}	u in l'	l'_2
z in l	l_2	z in r	r_{22}	0	r'

The kernel of the unification algorithm is depicted in figure 5.1. The algorithm has four cases. If the first term is a variable and if it has no binding we set the second term as its binding or if it has one we unify the term with the binding. When the second term is a variable we do the same with switched roles. If both terms have the same topsymbol we recur on their subterms. Else we have a clash. We use

the primitive functions variable?, binding?, binding, binding!, top, and sub as predicates, selectors, and modifiers for the data[iii]. It is clear that the lack of an occur check can be added in the same manner as the algorithm itself is stated. The explicit construction of the unifier is not necessary, it is delayed until the unifier is needed, that is, the critical pair is constructed. It is also evident that all other functions that are necessary for the Knuth-Bendix completion as for example overlapping, reduction, and ordering can be written analogously.

Figure 5.1: Kernel of the Unification Algorithm

```
(defun unify (term1 term2)
  (cond   (   (variable? term1)
              (if   (binding? term1)
                    (unify (binding term1) term2)
                    (binding! term1 term2)))
          (   (variable? term2) (unify term2 term1))
          (   (eq (top term1) (top term2))
              (every #'unify (sub term1) (sub term2)))
          (t  nil)))
```

The first step of the transformation is to represent the rewrite rules themselves as Lisp programs (see figures 5.2 and 5.3). It is obvious that the unification procedure in figure 5.1 works as before because for the unification the changes concern only the representation of terms which are hidden via data abstraction. The changes are local to the functions variable?, binding?, binding, binding!, top, and sub.

Figure 5.2: Associativity Rule as Lisp Program

```
(when ((λ  (l)
           (and (eq (top l) '+)
                ((λ (l1) (and (eq (top l1) '+)
                              ((λ (l11) (← 'x l11)) (sub 1 l1))
                              ((λ (l12) (← 'y l12)) (sub 2 l1))))
                 (sub 1 l))
                ((λ (l2) (← 'z l2)) (sub 2 l))))
       l)
  (bind '(λ  (r)
             (and (eq (top r) '+)
                  ((λ (r1) (← 'x r1)) (sub 1 r))
                  ((λ (r2) (and (eq (top r2) '+)
                                ((λ (r21) (← 'y r21)) (sub 1 r2))
                                ((λ (r22) (← 'z r22)) (sub 2 r2))))
                   (sub 2 r))))))
```

The advantage of the new representation is that rules are directly applicable to terms. For example the term $-a + a$ in the new representation can be reduced not by interpreting an abstract rule $-u + u \rightarrow 0$, but by applying the piece of Lisp code given in figure 5.3.

The function $\leftarrow$ checks whether the variable has a binding and if so whether it is the same as the new one to be set. If it has no binding it sets the new one, if the binding is not compatible it returns nil, that means that the match fails. In the right hand side of the Lisp rule the function $\leftarrow$ invokes an insertion of the binding which is executed by the function bind.

[iii] We use the standard naming convention for functional programming constructs of H. Abelson and G. Sussman [AS84]. The notation ((λ(l) body) input-l) of binding a new variable is also introduced there.

Figure 5.3: Left Inverse Rule as Lisp Program

```
(when ((λ  (l')
           (and (eq (top l') '+)
                ((λ (l'1) (and (eq (top l'1) '−)
                               ((λ (l'11) (← 'u l'12)) (sub 1 l'1))))
                 (sub 1 l'))
                ((λ (l'2) (← 'u l'2)) (sub 1 l'))))
       l')
  (bind '(λ  (r')
             (eq (top r') '0))))
```

We think this technique of encoding the match can be partially extended to full unification and hence can be used to implement a fast narrowing procedure. Of course this technique is not compatible with the theory unification algorithms, which have very dedicated structures.

By this way the procedure of applying a rewrite rule to a term t at toplevel is reduced to "(funcall rule t)", that is, customary matching can be omitted, it is encoded in the rule.

The next step in transforming the representation is to give the names of table 5.2 to all occurring subterms and to define them separately such that the rules are written just as in the lower part of the left hand sides of the figures 5.4 and 5.5. We introduce extra functions for lists of terms such that we do not need and-operators with different lengths.

The left hand sides of these figures show that all subterms look very similar and can be well organized in the computer memory. In a third step we can convert the Lisp functions into abstract assembly code which is shown in the right hand sides of the figures 5.4 through 5.5. In the sequel we shall often use the notion of "term l in Lisp notation" or "term l in assembler notation", but writing l in the usual way, for example "$x + y$ or $f(x, y)$ in Lisp notation".

The essential point is that the unification procedure still remains the same, again only the physical access to the subterms, binding, and type changes. We use "sub" with a numerical first argument if we access a single subterm and without this numerical argument for the list of all subterms.

The application of rules in the Lisp case can be handled by the Lisp interpreter and if a canonical system of rules is given by the corresponding compiler and the underlying machine. The pieces of Lisp code for the two rules of our example are given in figure 5.6a. An assembler analogon to the first rule is given in figure 5.6b.

5.2 The Transformation into Mnemonic Assembler Code

The transformation from "normal" terms to an "abstract" assembler language is done to ensure the soundness and completeness of the resulting proof procedure. In practice terms are created immediately in the final form. For the representation of replacement patterns in transformation rules we use the *reader macro notation* of Common Lisp: '(pattern ,expand pattern) constructs an expression where subexpressions preceded by commata are replaced by their values[iv].

[iv] For a detailed description of the notation see the books of G. Steele [Ste90] or P. Winston and B. Horn [WH89]. For example '(list ,x ,y 3) is equivalent to '(list 1 2 3) if x is 1 and y is 2.

Figure 5.4: Abstract Code for Term l

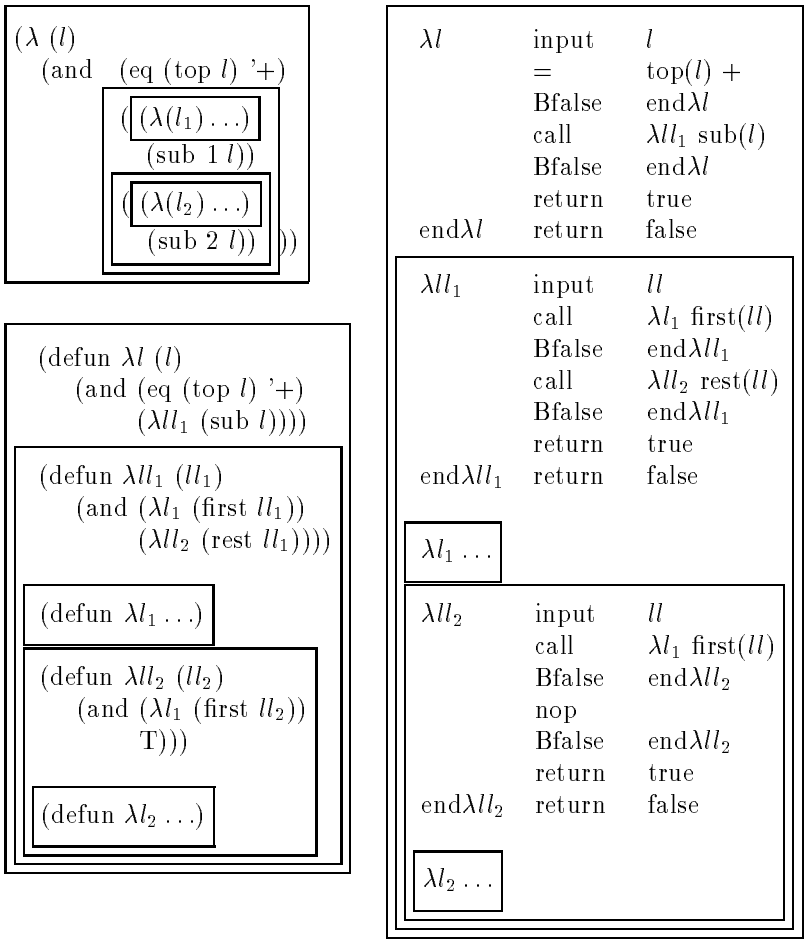
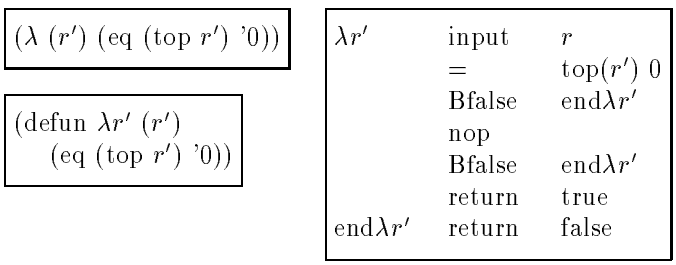


Figure 5.5: Abstract Code for Term r'



5.2.1 Abstract Code

We just give a brief and informal description of our goal language.

Each instruction is a line "{*label* }*id*{ *arg*}*{!*comment*}" with *label* a label identifier to be used in call or branch instructions, *id* the instruction name, {*arg* }* a list of arguments for *id* separated by blanks and !*comment* an arbitrary sequence of characters beginning with !, which serve as comment.

Figure 5.6: Rules

a) Two Rules in Lisp

```
(when (λl l) (bind 'λr))
(when (λl' l') (bind 'λr'))
```

b) Rule in Assembler

```
λrule     input  l
          call   λl l
          Bfalse end_int
          call   Bind λr
          return result
end_rule  return false
```

input	a	Inserts some input into register a.
=	$a_1\ a_2$	Checks whether some constants or register contents are equal and stores the result into a branching register.
Bfalse	$addr$	Jumps to the address if the branching register contains *False*.
call	$addr\ a_1 \ldots a_n$	Calls the routine $addr$ with the arguments $a_1 \ldots a_n$. There have to be n input instructions at the beginning of $addr$.
return	a	Leaves a program entered by call storing a into the branching register.
storage	#	Reserves # bytes of storage. This instruction reserves slots to label terms, which are used in superordinate algorithms.
true, false		Predefined constants.
top, sub		Primitive functions to access the function symbol and the arguments of a term.
first, rest		Primitive functions to separate term lists.

For each step of the transformation we have to consider three cases: compound terms, constants, and variables.

5.2.2 Terms → Lisp

The result of this step is depicted for our examples in the upper left corner of the figures 5.4 and 5.5.

Variable

For a variable the binding must be checked and set if necessary. This is done by applying the function ← described above.

$\mathcal{T}_\lambda(x) \quad \rightsquigarrow \quad (\lambda\ (t_i)\ (\leftarrow\ 'x\ t_i))$

Constant

The function for constants has just to do the clash check. It must return *True* if the constant symbol coincides with that of the input term. We use a primitive function top to access the top symbol of a term throughout all notations (see figure 5.5).

$\mathcal{T}_\lambda(c) \quad \rightsquigarrow \quad (\lambda\ (t_i)\ (\text{eq}\ (\text{top}\ t_i)\ 'c))$

Compound Term

The transformation for compound terms is done by recursion on the subterms. The first transformation rule separates the top symbol and performs the clash check and the handling of the list of subterms separately. For the access to the subterms we use a primitive function sub.

$$\mathcal{T}_\lambda(f(t_1,\ldots,t_n)) \quad \rightsquigarrow \quad \text{`}(\lambda\ (t)\ (\text{and}\ (\text{eq}\ (\text{top}\ t)\ \text{'}f)\ (,\mathcal{T}_\lambda((t_1,\ldots,t_n))\ (\text{sub}\ t))))$$

The second rule transfers an empty termlist, which can match all other lists because we pay attention to only call the resulting function on termlists of the same length.

$$\mathcal{T}_\lambda(()) \quad \rightsquigarrow \quad \text{`}(\lambda\ (tl)\ \text{T})$$

The third rule separates the first element and the rest of the list.

$$\mathcal{T}_\lambda((t_1,\ldots,t_n)) \quad \rightsquigarrow \quad \text{`}(\lambda\ (tl)\ (\text{and}\ (,\mathcal{T}_\lambda(t_1)\ (\text{first}\ tl))\ (,\mathcal{T}_\lambda((t_2,\ldots,t_n))\ (\text{rest}\ tl))))$$

5.2.3 Lisp → Single Functions

We have now Lisp code for all terms but terms are large interlocking objects. Our goal is to perform replacements by changing addresses and so we like to represent the logical composition of terms also physically and to represent all subterms as single objects. For these single functions in Lisp it is assumed that they all get distinct names. To assign the names to the function we use the Lisp special form *defun*.

In the example this is the step from the upper left to the lower left corner of the figures 5.4 and 5.5.

Variable

The transformation rule for variables just generates a name for the variable λ-expression.

$\mathcal{T}_l((\lambda\ (t)\ (\leftarrow\ \text{'}x\ t))) \quad \rightsquigarrow \quad (\text{defun}\ \lambda t\ (t)\ (\leftarrow\ \text{'}x\ t))$,

where λt is a new name.

Constant

The rule for constants works like that for variables. For an example see figure 5.5.

$\mathcal{T}_l((\lambda\ (t)\ (\text{eq}\ (\text{top}\ t)\ \text{'}c))) \quad \rightsquigarrow \quad (\text{defun}\ \lambda t\ (t)\ (\text{eq}\ (\text{top}\ t)\ \text{'}c))$

with λt again a new name.

Compound Term

The first transformation rule for compound terms handles the top level of the term, it generates one function for the top symbol and one for the subterms.

$$\mathcal{T}_\iota((\lambda\ (t)\\ \quad (\text{and } (\text{eq } (\text{top } t)\ 'f)\\ \qquad ((\lambda\ (tl)\ \ldots)\ (\text{sub } t)))))\ \rightsquigarrow\ \begin{cases}(\text{defun } \lambda t\ (t)\\ \quad (\text{and } (\text{eq } (\text{top } t)\ 'f)\\ \qquad (\lambda tl\ (\text{sub } t))))\\ \mathcal{T}_\iota((\lambda\ (tl)\ \ldots))\end{cases}$$

The second rule handles the empty termlist.

$$\mathcal{T}_\iota((\lambda\ (tl)\ \text{T}))\ \rightsquigarrow\ (\text{defun } \lambda tl\ (tl)\ \text{T}))$$

The third rule separates non-empty termlists. It generates one function for the separation, one for the first subterm, and one for the remaining termlist.

$$\mathcal{T}_\iota((\lambda\ (tl)\\ \quad (\text{and } ((\lambda\ (t)\ \ldots)\ (\text{first } tl)\\ \qquad ((\lambda\ (sl)\ \ldots)\ (\text{rest } tl))))\ \rightsquigarrow\ \begin{cases}(\text{defun } \lambda tl\ (tl)\\ \quad (\text{and } (\lambda t\ (\text{first } tl))\\ \qquad (\lambda sl\ (\text{rest } tl))))\\ \mathcal{T}_\iota((\lambda\ (t)\ \ldots))\\ \mathcal{T}_\iota((\lambda\ (sl)\ \ldots))\end{cases}$$

Here λtl, λt, and λsl are new names but they are also used in the corresponding substructure: λt in $(\lambda\ (t)\ \ldots)$ and λsl in $(\lambda(sl)\ldots)$. For an example see figure 5.4.

5.2.4 Single Functions → Abstract Code

For the example we step to the right hand sides of the figures 5.4 and 5.5.

Variable

An unbound variable just calls a SetBind instruction, which replaces itself by the term to be set as binding. SetBind always returns *True* when called non-recursively.

$$\mathcal{T}_\alpha((\text{defun } \lambda t_i\ (t_i)\ (\leftarrow\ 'x\ t_i)))\ \rightsquigarrow$$

λt	input	t
Bind	storage	Binding
	call	SetBind t
	Bfalse	endλt
	return	true
endλt	return	false

Variable with Binding

A variable with binding calls this binding instead of calling SetBind, hence the binding checks itself. We omit that SetBind called inside this λs-call is not allowed to set bindings but just has to check. This omission is for clarity of presentation.

λt	input	t
Bind	storage	Binding
	call	$\lambda s\ t$
	Bfalse	endλt
	return	true
endλt	return	false

Constant and Compound Term

For examples see the right hand sides of figures 5.4 and 5.5.

```
                                    λt      input   t
                                            =       Top(t) c
                                            Bfalse  endλt
(defun λt (t) (eq (top t) 'c))      ⇝       nop
                                            Bfalse  endλt
                                            return  true
                                    endλt   return  false

                                    λt      input   t
                                            =       Top(t) f
(defun λt (t)                               Bfalse  endλt
  (and (eq (top t) 'f)              ⇝       call    λtl sub(t)
       (λtl (sub t))))                      Bfalse  endλt
                                            return  true
                                    endλt   return  false
```

Term List

```
(defun λtl (tl) T))                 ⇝   nop

                                    λtl     input   tl
                                            call    λt first(tl)
(defun λtl (tl)                             Bfalse  endλtl
  (and (λt (first tl))              ⇝       call    λsl rest(tl)
       (λsl (rest tl))))                    Bfalse  endλtl
                                            return  true
                                    endλtl  return  false
```

5.3 Evaluation of an Implementation

For the evaluation of our concrete realization (that is given in more detail in appendix A) we chose three examples:

1. A group with the defining equations given in section B.3.4.
2. The central groupoid defined in section B.3.5.
3. A finite group (Z22) whose axioms are specified by the defining equations in section B.3.6.

For all experiments we chose a lexicographic recursive path ordering (LRPO).

We just implemented a pure Knuth-Bendix completion procedure for our experiments on compilation, because all problems occur already there. We present in the sequel the statistics for three different versions of the system: normal completion, completion with assembler data structure, and real application of the assembler procedures.

All times given here are computed on a SUN 4/370 and are statistical values in the sense that exact measurements are not possible for such small times. They are always given in seconds.

The loss of efficiency from "normal" to "new data structure" mainly has two reasons:

Table 5.3: Statistics Completion With Reduction of Critical Pairs (Times in Seconds)

Example	Group B.3.4	Groupoid B.3.5	Z22 B.3.6
Normal	0.79	3.05	49.34
New Data Structure	1.48	5.94	81.94
Compiled	1.32	5.82	73.48

1. It is due to the format of the "call" instruction of Sparc-assembler, where the address is not directly accessible for the user but must be shifted two bits. For the construction the sequence "01" must be added in front of the address to construct a legal "call" directive (see appendix A) from a given address:

   $$b_{31}b_{30}\ldots b_2b_1b_0 \rightarrow 01b_{31}b_{30}\ldots b_2$$

   With this coercion the access to substructures is more complicated than before.

2. The allocation of memory is more expensive than before, because all the instructions have to be written into the memory slots.

   Table 5.4 shows that the loss of efficiency is even present in that part of the procedure accelerated by the compilation.

Table 5.4: Statistics Completion, Only Matching, in Seconds

Example	Group B.3.4	Groupoid B.3.5	Z22 B.3.6
Normal	0.18	0.81	11.44
New Data Structure	0.32	1.16	13.18
Compiled	0.22	0.98	6.90

First of all we have to state that only about 25% of the time is spent for matching as can be seen in the table 5.5 and 5.6.

Table 5.5: Time Consumed by Subroutines in Percent (Normal)

Example	Group B.3.4	Groupoid B.3.5	Z22 B.3.6
Match	22.78	26.55	23.19
Unification	5.06	6.56	2.03
Memory Management	25.32	22.62	10.99
Ordering	0.01	0.02	2.47

Table 5.6: Time Consumed by Subroutines in Percent (New Data Structure)

Example	Group B.3.4	Groupoid B.3.5	Z22 B.3.6
Match	26.35	23.23	16.15
Unification	3.38	5.38	1.43
Memory Management	43.91	50.84	38.15
Ordering	0.01	0.01	1.54

Unfortunately the loss of efficiency caused by the new data structure is not offset by the compilation effect. Only the match time for large examples could be reduced as can be seen in the fourth column of table 5.4.

For these numbers we can conclude that compilation does not really disturb the efficiency of the completion process but also does not contribute to more efficiency. That means if a system is employed that intensively uses reduction and only sometimes a completion step is slipped in then the presented method can be used to get completion together with a fast reduction. In chapter 4 we showed that heuristics for choosing the next critical pair have a more drastical effect on the efficiency of the completion procedure.

To repair the deficiencies of our compilation approach we propose to take the following steps:

1. The amount of time used for memory allocation can be reduced when the routines are shorter. Using a more stack oriented architecture of our system (like the WAM) could reduce this size. But it is not dedicated to the standard computer architecture.
2. The same method as in 1 would also avoid many of the address manipulations mentioned above.
3. Using an underlying machine with a more homogeneous address format also would reduce these manipulations.
4. The usage of dedicated memory management routines.

5.4 Unfailing and Conditional Rewriting

The incorporation of standard extensions to the Knuth-Bendix procedure does not cause additional problems, however, we did not implement these extensions because we do not think that there is a considerable enhancement even comparable to that in section 5.3.

5.5 Possible Refinements

5.5.1 Compiled Unification

Prolog derives part of its speed up from the compiled unification process. In table 5.5 we gave the percentages of time used for the unification in our compilation examples. These times are smaller than those for matching and therefore it is less promising to get an enhancement. So we renounced on any experiment with compiled unification. In section 5.5.2 we cite our results of unifying indexing trees, but this is only useful because the trees have some other advantages.

5.5.2 Indexing Trees

Another approach to an equality reasoning machine can be developed based on interpreting indexing trees.

All non resolution relevant equality operations can be performed on the term level of indexing trees. They can be used for the representation of reduction rules and terms as well as a basis for the compilation of reduction and other special algorithms like narrowing. In this case a two step compilation as given in figure 5.7 can be performed.

A compiler for term reduction systems into Lisp is described by S. Kaplan [Kap87], an implementation in our system works very similar but explicitly uses a layer between simple clause rules and Lisp code, which allows to incrementally construct more efficient code.

Figure 5.7: Compilation

The two tranformation steps:

- rules, clauses → abstraction tree
- tree → Lisp code

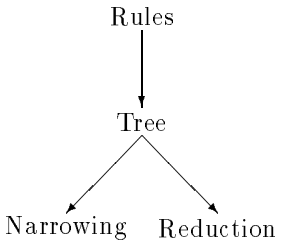
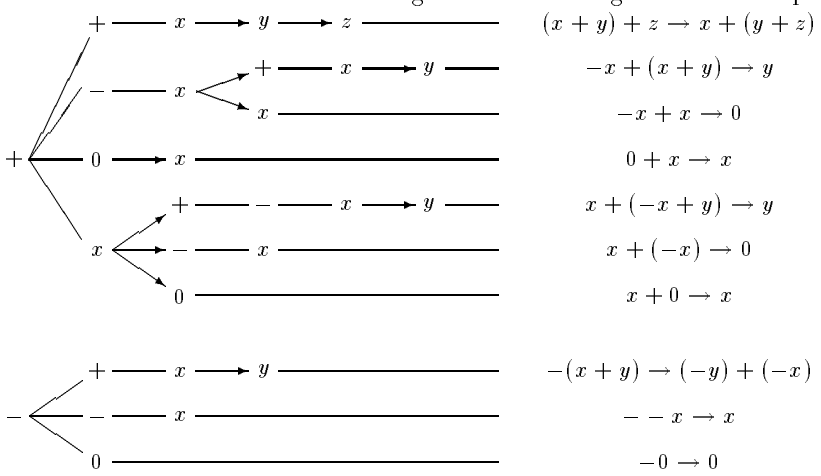


Figure 5.8: Indexing Tree for Group



In figure 5.8 we choose an example for an indexing tree as they are used in the system. It represents the complete set of reduction rules for groups. The leafs of the tree are labeled with the rules, the paths to the leafs represent the flattened left hand sides, the internal nodes are labeled with operator symbols and variables, arrows are drawn for the recurrence on higher levels of terms, that is, after variables or constants that are leafs in the term tree. They are not really needed but make the figure more readable.

These trees are a special sort of indexing trees [WOLB84] or abstraction trees [Ohl89], but we use them as the programs of an abstract machine. An interpreter for this machine will be described below (figure 5.9). A. Nonnengart used such directed acyclic graphs [Non86]. He evaluated run times for unification processes on such trees that coincide with our times for rewriting given in table 5.7.

For the compilation process into Lisp each subtree for an operator symbol will produce one Lisp function [Kap87].

The figure illustrates that these trees are well suited for parallelization (see [HM89, Den91]). For an implementation of this system two abstraction levels can be distinguished. The first step of the compilation can be performed using a special interpreter for the indexing trees when performing a reduction or both steps can be performed finally using compiled Lisp code for the reduction. An interpreter for indexing trees is depicted in figure 5.9. The problem with the second version is that in the case of theorem proving the system of reduction rules changes dynamically causing changes of code that are almost impossible to be handled in compiled Lisp. Interpreted Lisp, however, is very slow. Using a similar technique as at the begin of the chapter will lead to comparable results because the distance between C and compiled code is smaller than between Lisp and code.

We now try to discuss the problem when to use which of the two levels by evaluating the statistics for our implementation.

For the performance test we selected the following examples:

1. $(((((-x+-x)+-x)+-x)+-x)+(((((x+x)+x)+x)+x)+x)=0$ in a group (standard example)
2. $(((((((((x+x)+x)+x)+x)+x)+x)+x)+x)+x = x+(x+(x+(x+(x+(x+(x+(x+(x+x))))))))$ in an associative structure with + (small set of rules, in this case one rule)
3. $(((x+1)(x-1)^2)(x+1))(x-1) = x^5+(-(x^3)+(-(x^4)+(x^2+(x^4+(-(x^2)+(-(x^3)+(x+(-(x^4)+(x^2+(x^3+(-(x)+(-(x^3)+(x+(x^2-1)))))))))))))))$ in a ring (more rules)

All of these examples work with a completion process and succeeding reduction using some of the newly generated reduction rules. We employed the MKRP-version of our system, hence the runs were relatively slow.

In table 5.7 we collected the running times of the pure final rewrite steps, in table 5.8 the complete refutation times including the completion process are given. We fed the system with a minimal set of axioms.

Table 5.7: Running Times for Rewriting (in Seconds)

Example	1	2	3
pure rules	1.85	0.09	305.83
interpreted trees	0.17	0.03	9.97
compiled procedures	0.04	0.02	2.37

The entry "pure rules" means that the rewrite rules are not compiled in any sense, they are used completely independent, "interpreted trees" means that trees as those described above are successively constructed and maintained, "compiled procedures" means that these trees are compiled into Lisp. In the second table the last point is changed to "always compiled", which means that after each change of the tree, new Lisp code is generated. Another possibility would be to compile only if a canonical system is derived, that is, all critical pairs between unit equations are confluent. We omitted this case because for the considered examples this coincides with the "interpreted trees" case. In addition we tested the system with the compilation of pure rules depicted in the line "pure rules compiled".

Table 5.8: Running Times for Refutations (in Seconds)

Example	1	2	3
pure rules	8.84	4.32	473.14
pure rules compiled	9.82	6.00	302.67
interpreted trees	8.68	4.40	292.25
always compiled	35.79	9.39	640.13

The last step needs the longest time because equality examples are very dynamic and the final reduction and resolution operation is performed just after the last rule is derived or even it needs not to be derived to obtain the result.

A Lisp procedure for the reduction using indexing trees is depicted in figure 5.9. It works by stepping recursively through a given termlist and a list of indexing trees setting bindings for variables in the tree. The initial input are lists with one term and one tree. If the termlist is worked off we have successfully reached a leaf of the indexing tree and return the stored right hand side with inserted bindings. If there remain terms to be reduced we select a subtree via the "some" statement with variable or fitting function symbol. In the function case we have to consider the subterms, in the variable case we have to check and set the binding.

The algorithms for indexing trees can be refined to work for special sorts of narrowing like commutation narrowing and their specialized reduction rules.

Figure 5.9: Interpreter for Indexing Trees

```
(defun reduce (termlist trees)
  (if (null termlist)
      (bind (stored-tree trees))
      (let ((first-term (first termlist)))
        (some #'(lambda (tree)
                  (let ((symbol (symbol tree)))
                    (cond ((equal symbol (top first-term))
                           (reduce (append (sub first-term) (rest termlist))
                                   (subtrees tree)))
                          ((variable? symbol)
                           (cond ((not (binding? symbol))
                                  (binding! symbol first-term)
                                  (reduce (rest termlist) (subtrees tree))
                                  (reset-binding! symbol))
                                 ((equal (binding? symbol) first-term)
                                  (reduce (rest termlist) (subtrees tree))))))))
                trees))
```

A. Nonnengart described a representation of terms with acyclic directed graphs [Non86] which is very similar to our indexing trees. He also has the result that unification on graphs is about ten times faster than the usual one, but we have to keep in mind that the unification process just takes about 5% of the time consumed by a typical completion procedure (see table 5.5).

We can recapitulate at the end of this chapter the results of table 5.5. None of the operations in the completion procedure is so dominating[(v)] that its speed-up can result in a considerable speed-up of the whole process, hence compilation seems to be no promising approach to completion, because it will always fasten just one of these components.

From our compilation experiments in Lisp (see table 5.7) we expected an essential speed-up when using our compilation technique. But implementing the system in C led to a speed-up that outperformed any possible speed-up by compilation.

[(v)] Unification $< 5\%$, matching $< 25\%$, ordering $< 5\%$.

Chapter 6

Aspects of Integration

The previous chapters presented various equality reasoning methods as developed in the last decades (chapter 2), some enhancements of our own (chapter 3), experiments with various heuristics (chapter 4), and our experiences with the development of a compilation approach to rewriting (chapter 5). That is, we have collected a large equality reasoning toolbox.

This chapter sketches how we obtain a powerful integrated system from the equality reasoning toolbox. It contains a presentation of a system architecture for such an integrated equational theorem prover, and the need for different inference systems to prove special problem classes is discussed. A point of interest is, of how special features of the theorem proving programs developed in our research group fit into the superposition approach, including compilation and heuristics (see sections 6.4 through 6.6).

The MKRP-system was first of all used as a general scratchpad for our ideas. Furthermore the integration into this theorem prover makes available a pile of features that were developed over many years [OS89]. For our experiments on theory completion we used the unification algorithms collected and implemented in the HADES tool box [Sch89, Prä92a]. In order to represent the equational proofs in a human-oriented way additional rules were integrated into the proof presentation system GENTZEN [Lin90].

6.1 Rules for Building Theorem Provers

Equality is an indispensable feature for a powerful theorem prover. The integration of equality reasoning showed that certain rules must be followed, when building a general theorem prover if the integration of a basic feature like equality is desired. A comparably basic concept would be sorts.

Of course it is necessary to heed all basic rules of software engineering such as data abstraction or documentation issues. In the scientific area it is also necessary to keep the systems as small as possible to make them usable for someone outside. This is a big advantage of the programming language C over Common Lisp. Another suggestion is to always project two implementations from the very beginning, the first one a prototype without too much emphasis on performance issues, just for learning concepts and making experiences (rapid prototyping). The second implementation should be done by the same persons as the first one and integrate all known and desired features. You should never believe that you can program something you do not know, your first errors will survive your presence in the project team! Hence rapid prototyping is worthless if it is not followed by a reimplentation.

The first and main rule for implementing theorem provers is to never invent any complicated things, the main parts of theorem provers must be simple and transparent. It never pays to complicate matters, may they reside in the inference rules, the control strategy, or the data structures.

For the inference system this means that we prefer a simple homogeneous set of calculus rules with ideally one rule as a basis which incorporates all facilities necessary to obtain an efficient theorem prover. This includes also a simple control for the selection of possible steps.

Rule 1	*Use a simple and homogeneous basic calculus!*

Example 6.1 (Rule 1) *The superposition calculus unifies all important inference rules in contrast to a natural deduction calculus which has several inference rules that are controlled differently. Resolution and paramodulation can have different heuristic functions and there exists no correlating function that can be used in general (see also the strategy* H-C *in section* 4.2.1*).*

Often complicated data structures as for examples "clause graphs" mix data and control information. In clause graphs the potential resolution steps which must be considered for the control of the theorem prover are encoded as links in the graph which is in fact a data structure. When selecting a step, the control must consider all links to decide which should be used. This makes it necessary to keep all links additionally in a list or always compute this list, which must be also updated when inheriting the links after a resolution step. Then the actual state of the system becomes very confusing, because relevant information is kept more than once. The more complicated the structure the more complicated is it to hide this structure from the control module and hence from the user.

Simple data structures cause more flexibility and portability:

- A function to select one of a set of equal objects can be changed very flexible and adapted to many special cases. To select one out of a selection of very different objects is very difficult.
- Basic constructs can be the same for various algorithms if they are sufficiently simple.

Of course it is possible to translate the basic data structure of one algorithm to another but especially in the case of theory unification this causes a lot of overhead.

If you begin the implementation with a complicated system, it is almost impossible to integrate anything because it must be in tune with all other already existing components. The problem can be further complicated if various components use different data structures and are differently controlled. This contemplation leads to the following two rules.

Rule 2	*Use simple abstract data structures!*

Example 6.2 (Rule 2) *Lists of clauses are more suitable for other algorithms than clause graphs. Normalization, for example, may it be by rewriting or otherwise, can be handled in clause graphs only by reconstructing the incident links of the normalized clause node, hence it is necessary to know the link concept when normalizing clauses. If two algorithms use functions to create terms then these two functions should differ at the most in the name.*

The simple data structure makes it easy to adapt other programs also using simple data structures.

Example 6.3 (Rule 2) *Two main problems we had when we integrated external programs were firstly two different complicated data structures for* HADES *and* MKRP *and secondly the lacking abstraction in the implementation of the AC- and AC1-match implementations* [Hof88].

It is typical for AI programs that some of their modules compute a lot of information to be used by other parts. This causes a fundamental communication problem. It is evident that all stored control information should have the same representation and format. Too much communication causes enormous overhead.

Rule 3	*Use a homogeneous format to represent control information!*

Example 6.4 (Rule 3) *Use homogeneous, that is comparable, heuristic values for different proof steps.*

Neglecting this rule makes it normally infeasible to use foreign external programs for additional inference rules directly. It is also often the case that the control of such external programs is implemented completely different. To cope with such situations we need the following rule.

Rule 4	*Use only external ideas, not external programs!*

Example 6.5 (Rule 4) *We do not use an external implementation of the Knuth-Bendix algorithm, but simulate it within our calculus.*

A related question is which control information should be stored at all. Often there is the problem that programs are mainly occupied with managing and updating control information. Hence it is necessary to keep this information as simple as possible. We can distinguish dynamic and static information. It is no problem to store and use static information which never changes. If the information changes with each step it may be advisable to recompute it whenever it is needed, rather than to keep this information stored and to update it permanently.

Rule 5	*Store static information, recompute dynamic information!*

Example 6.6 (Rule 5) *The heuristic value for possible results can be stored. It is not advisable to store too much indexing information for finding terms, because its update can be very expensive.*

Often dynamic information can be approximated by static information. For example, we can use old heuristic values if it is too expensive to recompute them. We can use old narrowing trees, even if new rules are added (see section 4.1.3) or we can omit the normalization of possible results, that is we do not keep critical pairs reduced (see section 4.2.4).

For integrating additional incompatible inference rules these rules should be not visible on the top level of the inference system. Their results are only computed up to a certain level and special data structures as for example continuations can be used to store the results. The computation of this information can be interleaved with the main loop. These steps should only be used if they can be combined to abbreviations on the main inference level.

Rule 6	*Abbreviate additional incompatible inference rules!*

Example 6.7 (Rule 6) *Narrowing can be used to simulate E-resolution; then one E-resolution step which behaves like a normal superposition step corresponds to a couple of narrowing steps (see section 3.1). The terminator can be used to find unit refutations or unit clauses as lemmata.*

This technique minimizes the selection problem for the next step. There must be no distinction between different rules, the result is of the same type at the top level of the inference loop.

Of course the amount of stored information can be extended enormously by such rules, but the information is not relevant and hence cannot confuse on the main inference level. There the information should be as narrow as possible.

Rule 7	*Keep the actual set of formulae and the formulae themselves small!*

Example 6.8 (Rule 7) *In resolution theorem proving all available reduction rules like subsumption, tautology removal, and replacement resolution should be used. Using sorts may be also a technique to keep the formulae small. Not using resolution is for special problems a good technique to keep the set of formulae small. This enforces to use several theorem provers.*

Now we come to the rules that are more specific for equality. For equality, rule 6 is not valid because equality is too basic to be handled this way. It is overridden by the following rule.

Rule 8	*Never handle basic things independently from the basic calculus!*

Example 6.9 (Rule 8) *Superposition embeds equality and resolution into one step, E-resolution simulates equality as an abbreviation and is therefore unsuitable to cope with equations in general.*

Hence equality reasoning should always be fully integrated and should never be used as a preprocess or as a support module. Using first the Knuth-Bendix algorithm to compute new equations and then use these equations and ignore all further equational steps would be such a preprocess.

Because equality works on the term level we need a specialization of rule 7.

Rule 9	*Keep terms small!*

Example 6.10 (Rule 9) *Use rewriting, that is, normalization of terms wherever it is possible.*

The summary rule for us would be: whenever equality is present use superposition and rewriting!

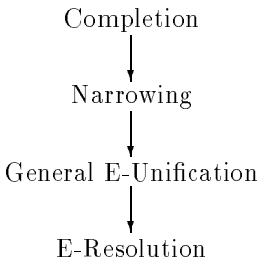
6.2 Control Through System Architecture

In this thesis we have elaborated so far mainly the details of various equational reasoning tools rather than an explicit global supervisor to control the various components. Now we shall take a more global view on the system architecture. In this section we focus on tools that are compatible with the superposition calculus in the sense that they use essentially the same inference rules. The next section will concentrate on inference rules which are incompatible to our approach. First we can distinguish strategies selected explicitly by the user or the system and strategies that are inherent[(i)] to the calculus.

The explicit selection can be organized in such a way that the general strategies reduce to special algorithms if available. The decision how to refine the strategy must be made globally. For example, if it can be detected that narrowing is complete (that is, a canonical reduction system exists) general equality reasoning is reduced to a pure narrower with the best possible narrowing strategy according to the equational theory as depicted in figure 6.1. Some narrowing strategies are only applicable if additional restrictions are imposed on the equational theory. If narrowing is not complete it is only applied in a lazy and heuristic way.

Figure 6.1: Combination Completion – E-Resolution

An arrow $A \rightarrow B$ means that A is used to implement B.



As a second example we mention the compiling version of the system. It can only be used if theory unification is not necessary and that must be decided globally by taking into account special declarations of function symbols. Narrowing also can only be reduced to a compiled version if no E-unification is necessary. This selection depends on the compilability of the theory unification algorithm.

If no algorithms are necessary the calculus itself may reduce to a new one. An example for such a local reduction of the calculus is from general superposition to Knuth-Bendix completion if only unit equations are present. This reduction is inherent to the system.

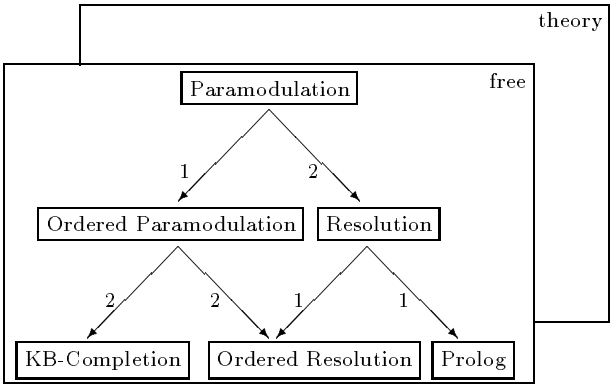
Some of the possible refinements are depicted in figure 6.2. A step labeled with 1 is always done by applying a restriction to the inference rules, in the paramodulation case using a reduction ordering, in the Prolog case using SLD-resolution. A step labeled with 2 is automatically embedded into the calculus. The completion case happens when only unit equations are present, the resolution case when no equations are given at all.

The introduction of theory unification is possible for all items in the figure leading to a picture of the same shape with an "E-" before each symbol, for example E-resolution or theory completion. We will discuss the suitability of this theory operator for completion in section 6.5.

[(i)] We very freely use the notion "strategy". Normally a strategy is something to control an inference system and the notion cannot be used without relating it to this system. But of course one can simulate strategies of one inference system in another one. this justifies our open use of the term "strategy". For this problem see also our distinction of restriction and selection strategy in section 4.2 on page 44.

Figure 6.2: System Architecture

An arrow $A \rightarrow B$ means that B is a subcase of A (a restriction of the inference rule).



6.3 Global Supervision

Now we come to the possibilities to integrate methods that are incompatible with the proposed superposition calculus as for example decomposition based equality reasoning [Blä86] or classical induction theorem proving [BM79]. The incompatibility is mainly caused by completely different inference rules.

Firstly the question must be answered whether all facilities have equal rights or whether they are invoked in an hierarchical order.

As argued in section 6.1 we prefer an hierarchical approach using a simple basic inference rule and embedding the other rules on a subordinate level which is invisible in the main inference loop. Hence global supervision of the classical type is not necessary at all.

An "equal rights" approach would be to have several theorem provers available and to decide either manually or automatically which should be used for a concrete problem. In the Ω-MKRP project [Sie92] this selection will be embedded into a dialogue oriented natural deduction theorem prover which is controlled manually at the first development level. For an automatic selection we have to answer the question whether and above all how the given formulae help to decide which prover to use.

There are some general approaches to classify equality reasoning methods. One of them distinguishes between term replacement and difference reduction methods [Blä86] (see section 2.4, page 5). Term replacement works by substituting terms using equations, difference reduction considers at least two terms and tries to make them equal by inserting equations at top level and proceeding recursively on the subterms for terms with the same function symbol.

The term "difference reduction" is used at two levels of abstraction. First it denotes term decomposition as explained above. This is at the same level as term replacement. Secondly it means that the whole approach operates on a higher (AI) level and reduces semantical differences [Blä86]. We think that the second level is not adequate to control operations on the term level as elaborated in section 2.4 on pages 10 and 14.

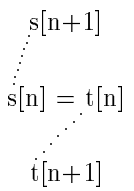
This classification is related to a classification of the axioms and theorems containing the equality predicate. It concerns the structure of equations with regard to similarity of their left and right hand sides, and with respect to similarities between the whole equations. Commutativity for example is an axiom, which is itself structured in the sense that its left and right hand side are very similar. It is very handy for difference reduction. The single axioms for left and right zero however are examples for formulae without such a structure. They are well suited for replacement style reasoning. Corresponding properties can be attached to theorems.

Additionally theorems can be classified according to their relation to axioms. Especially induction theorems possess a typical structure and a strong relationship to their hypotheses (for example $x(y+z) =$

$xy+xz \Rightarrow (x+1)(y+z) = (x+1)y+(x+1)z$) [Hut90, BvHSI90]. The structure of the induction scheme is depicted in figure 6.3 using a scheme of an equality graph (see section 2.4.2). A comparison of this type of structure usage in induction theorem proving (named rippling) with an inductionless induction prover based on completion is given by R. Barnett, D. Basin, and J. Hesketh [BBH91].

Figure 6.3: Induction Scheme

This equality graph depicts a scheme for the induction step of the theorem $\forall n : s[n] = t[n]$.



Another dimension for the classification is whether there is just one theorem or several theorems. Many theorems can occur when equality is embedded in a resolution prover via E-resolution, namely one for each pair of literals with the same predicate and opposite sign.

This classification induces corresponding reasoning methods. Structure in the axioms induces the usage of a decomposition approach, structure in the theorems induces special transformation methods [Hut90] based on difference reduction. Many theorems in combination with structured axioms induce a graph based decomposition method to store partial solutions to be shared such that they can be used at different positions.

In the unstructured case rewriting and narrowing bring the literals to normal form, and then they are unifiable.

It is interesting to remark that the best fitting method to solve equality reasoning problems can also serve to classify them. We could name this phenomenon statistical classification and the result corresponds to the structural classification mentioned above.

Very often the really complex replacement problems are characterized by orientable equations.

When equations occur together with other predicates the problem is to determine which constituent is responsible for the main work. In the case where resolution does the main work we have simple replacement problems and it is often better to use difference reduction. In the other case where paramodulation is the more frequently used operation we have complex replacement problems with orientable equations and only a few narrowing problems occur.

If just one unorientable equation is present often a theory unification algorithm exists to solve this equation, or narrowing with a canonical set of rewrite rules is the method of choice.

Hence the type of reasoning in an "equal rights" approach could be selected problem driven, that is, calculus and inference rules are selected automatically when appropriate clauses are present.

Additional aspects of global supervision are the use of analogy [Ker89] and plan generation [Bun89]. In contrast to the other control components mentioned in this section analogy and plan generation do not consider just one example, but a whole set of solved problems, which are for example stored in a mathematical data base.

They need methods to detect similarities of the given problem to neighboured problems stored in the data base. They also need a method to store the proceeding of finding a special proof. For this purpose tactics can be used [KP92]. One day such automatically generated tactics could represent something like contemplation and analysis of computed problems.

6.4 Integration into an Existing Theorem Prover

In the chapters 3 and 4 we gave the main rules of how equality reasoning features can be integrated into a resolution theorem prover. Embedding equality handling into an existing resolution based theorem prover as described in chapter 3 would be worthless unless useful properties of this theorem prover could be inherited.

In this section we list some facilities of the MKRP-system that are retained by our method of integrating the equality reasoning mechanism.

Splitting, for example, makes it possible to formulate some problems in a more natural way, because the parts of the proof are proved separately. A typical application of this simple but nevertheless powerful method is the break-up of an equivalence $A \Leftrightarrow B$ into two separately provable implications $A \Rightarrow B$ and $B \Rightarrow A$. This is of great value especially for theorem provers based on clause normal forms. The clause sets can be drastically smaller using splitting. Example 53 of F. Pelletier (see section B.2.5, page 97) has 32 clauses each with six literals without splitting. With splitting two splitparts each with only eight three-literal clauses must be refuted. Another example for this mechanism is 6.11.

Example 6.11 (Splitting)
The three parts of a circular implication can be given as one formula and are then proved independently. (The statements are equivalent in a group.)

1. $x + y = y + x$
2. $(x + y) + (y + x) = (x + x) + (y + y)$
3. $-(x + y) = -(x) + -(y)$

Another distinguishing feature of the MKRP theorem prover are sorts [Wal84, Sch88]. They are compatible with our approach as long as we take care that the ordering is compatible with the sort structure. Of course there is an abundance of literature describing the combination of sorts and equality reasoning methods. J. Goguen, J.-P. Jouannaud, and J. Meseguer [GJM85], A. Dick [Dic85], K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer [FGJM85], M. Bidoit and M. Glaudel [BG85], and J.-P. Jouannaud and P. Lescanne [JL87] take a view from program specification on rewriting and sorts, the viewpoints of R. Cunningham and A. Dick [CD85], J. Goguen and J. Meseguer [GM85], G. Smolka and colleagues [SNMG87], as well as the works of J. Gallier and T. Isakowitz [GI88], and M. Schmidt-Schauß [Sch88] do not regard such a concrete application. Especially [Sch88] must be considered when fully integrating sorts and rewriting. An open problem in this area is that it is not clear how strong the restriction on the correlation of the reduction ordering and the sort hierarchy must be.

Another way to introduce sorts in a theorem prover in contrast to the static programming language oriented way is their dynamic integration [Coh87, Rug91, Wei89, Wei91] in form of sort literals like $\in(Tweety, Birds)$. This opens the perspective to define richer structures by describing sorts not only using constants but also compound terms. In this case equality can be easily incorporated via superposition into sort terms which are handled as normal literals. Sorted unification usually works by restricting the unsorted unifiers by some constraints. For order-sorted unification the codomain term must have the same or a subsort of the domain variable. For more complicated sort structures the restrictions become undecidable and to make the unification terminating residue literals must be added to the resolvent. They represent the unfinished decision computation. It is unclear up to now which restrictions on the unification algorithm must be imposed if equality is combined with dynamic sorts as mentioned above [Wei93].

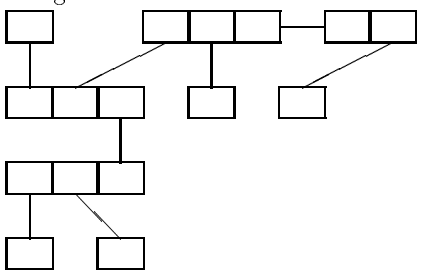
One of the most powerful tools in MKRP is its dedicated clause graph reduction facility [Prä85, EOP89]. Especially the subsumption rule is very useful in the equality reasoning context. We implemented an extended form according to chapter 3, which allows a lookahead of demodulation steps such that a lot of unnecessary link generations could be suppressed. But there remains an enormous overhead caused by the generation of these links.

Another source of the strength of the MKRP-system is the "terminator" which is a tool to find unit refutations. Terminator situations are like the one displayed in figure 6.4. This connection graph has

no cycles in contrast to arbitrary connection graphs. Such situations are searched beginning with unit clauses (see [AO83]).

In another mode the terminator program searches unit clauses which are often very useful as lemmata in resolution theorem proving. In this mode the terminator performs forward search for resolution like the completion procedure for equations. Therefore completion together with rewriting can be regarded as a terminator for equations. Of course this is only one aspect: completion is fully integrated into the calculus whereas the terminator is a distinct program interleaved with the resolution rule. Hence in this point the terminator is more comparable with the method how narrowing is integrated (see section 4.1).

Figure 6.4: Terminator Situation



In table 6.1 we list the methods available in MKRP that are used to solve F. Pelletier's examples [Pel86]. We used the B-G strategy. With other strategies we get better quotients *proof steps / made steps* but almost the same computation times. For the examples 51 and 52 we used the ordering of 51a in table 4.1.

Table 6.1: Pelletier: Statistics and Features

Example	Proof Steps Including Rewrites	Made Steps	Time	Features
48(B.2.1)	9	9	6	
49(B.2.2)	15	32	13	
51(B.2.3)	17	23	33	
52(B.2.4)	17	23	33	
53(B.2.5)	564	1434	2137	splitting, finite domain, ordering
54(B.2.6)	22	24	31	terminator
55(B.2.7)	15	29	22	
56(B.2.8)	15	15	9	
58(B.2.9)	2	3	5	
61(B.2.10)	3	3	3	
63(B.2.11)	15	45	18	
64(B.2.12)	7	10	10	
65(B.2.13)	8	14	11	
73(B.2.14)	195	215	5929	

6.5 Usage of Theory Unification

As mentioned earlier simple completion and demodulation alone are not the unique mechanism to solve the problem of handling the equality predicate automatically. The first implication ($1 \Rightarrow 2$) of example 6.11 is a suitable instance of such a problem. To prove 2 nothing must be done, except to first switch

the middle pair of identifiers and then the right pair. But associativity can only be applied left to right and so we have an equation $x+(y+(y+x)) = x+(x+(y+y))$ and a relatively complicated proof using unfailing completion with 26 completion and demodulation steps is generated by our theorem prover. Using AC-unification the proof consists of a trivial unification step, which only has to state that the two terms are equal. This shows that theory unification is a powerful tool in combination with rewriting.

The HADES-system (**H**ighly **A**daptable **De**duction **S**ystem) was designed as an environment to develop inference systems and unification algorithms [OS88, section 7], [Prä92a]. Theory unification [Tep89] is the only feature of the HADES-system used in our context.

Unfortunately the universality of the approach to integrate all available features into one system is broken. None of the theory unification algorithms works with our compilation approach.

Most of the theories are incompatible with the link inheritance mechanisms. They require that all links are newly constructed, hence in this case a non clause graph approach should be better.

We have a prototype implementation [Mah91] of such a system to evaluate the usability of unification algorithms and disunification for the completion method. Furthermore theory unification algorithms require special data structures to work efficiently.

Using just a special unification algorithm for the commutativity axiom leads nowhere because for commutative function symbols an ordering respecting the symmetry is necessary and then the associativity axiom is not directable. And commutativity without associativity does rarely occur, anyway.

Often associativity, commutativity, and an identity element occur together and one has to decide which unification algorithm should be selected to overcome the problem to be proved: AC- (associative and commutative) or AC1-unification (AC with identity). For the discussion of their properties we first try to elaborate the structure of the occurring unifiers. Imagine our task would be to unify the terms $f(x,y)$ and $f(u,v)$. They have the unique most general unifier $\{x{\leftarrow}u, y{\leftarrow}v\}$ for the empty theory as well as for associativity alone. For AC1 and AC intuitively there should be two most general unifiers: $\sigma_1 = \{x{\leftarrow}u, y{\leftarrow}v\}$ and $\sigma_2 = \{x{\leftarrow}v, y{\leftarrow}u\}$. But the usual AC1 algorithm works differently. In the case without free constant or function symbols it is always possible to compute a unique most general unifier. For our task the result is $\sigma = \{x{\leftarrow}f(v_1,v_2), y{\leftarrow}f(v_3,v_4), u{\leftarrow}f(v_1,v_3), v{\leftarrow}f(v_2,v_4)\}$ with new variables v_1, v_2, v_3, and v_4, which is more general than both, σ_1 and σ_2. For example $\sigma_1 = \sigma \circ \{v_1{\leftarrow}u, v_2{\leftarrow}1_f, v_3{\leftarrow}1_f, v_4{\leftarrow}v\}$ with 1_f the identity element corresponding to f. The crossing of commutativity is encoded into the structure of the unique unifier. This phenomenon is typical for AC1-unification and one can imagine the increase of complexity with growing number of variables.

The set of AC-unifiers is always computed by instantiating variables in the codomains of AC1-unifiers with the identity element, yielding

$$\begin{aligned}
\tau_1 &= \{x{\leftarrow}f(v_1,v_2), y{\leftarrow}f(v_3,v_4), u{\leftarrow}f(v_1,v_3), v{\leftarrow}f(v_2,v_4)\},\\
\tau_2 &= \{x{\leftarrow}v_2, y{\leftarrow}f(v_3,v_4), u{\leftarrow}v_3, v{\leftarrow}f(v_2,v_4)\},\\
\tau_3 &= \{x{\leftarrow}v_1, y{\leftarrow}f(v_3,v_4), u{\leftarrow}f(v_1,v_3), v{\leftarrow}v_4\},\\
\tau_4 &= \{x{\leftarrow}f(v_1,v_2), y{\leftarrow}v_4, u{\leftarrow}v_1, v{\leftarrow}f(v_2,v_4)\},\\
\tau_5 &= \{x{\leftarrow}f(v_1,v_2), y{\leftarrow}v_3, u{\leftarrow}f(v_1,v_3), v{\leftarrow}v_2\},\\
\tau_6 &= \{x{\leftarrow}u, y{\leftarrow}v\}, \text{ and}\\
\tau_7 &= \{x{\leftarrow}v, y{\leftarrow}u\} \text{ in our example.}
\end{aligned}$$

It can be concluded that the set of AC-unifiers has the advantage that smaller unifiers can be heuristically selected whereas for the theory AC1 the system is forced to use the most complicated one. We could call the phenomenon as "conservation of problem complexity" in the solution whatever way it is computed.

M. Stickel [Sti84] as well as D. Kapur and H. Zhang [KZ89] show that AC-completion is useful but only with specific tricks which are not valid for Knuth-Bendix theorem proving in general: the cancellation law for the additive group (group specific), special weighting functions for the selection of critical pairs (example specific, see also section 4.2.3), various possibilities to detect symmetries in critical pairs, unifiers, or even superpositions (commutativity specific), "systematic instantiation" which is a rule derived from T. Wang's Z-module reasoning for non-associative rings [Wan88] (ring specific). In

addition "the blocked superposition criterion" showed to be very useful in this case. It states that critical pairs can be omitted if the corresponding unifier can be reduced by one of the parent equations.

There are three levels to constrain the construction of critical pairs. The first concerns the positions where superpositions take place. Some of the constraints enforce changes to the theory unification algorithm itself because the construction of unifiers must be restricted as early as possible to be more efficient. The third level allows to omit ready unifiers and critical pairs.

Another essential aspect when selecting the unification algorithm is that orderings are stronger for AC- than for AC1-completion because of the presence of collapsing axioms in AC1-theories. For AC1-completion the two disadvantages of exploding structure and weak ordering accumulate.

The effect can be best depicted with one of the simplest critical pairs computed in the completion process of an Abelian group: $x+y = z+y-(u+z-(u+x))$. This equation is selected to be directed but this is not possible due to the restrictions on the ordering. These are necessary because, for example, the instance obtainable when applying the unifier $\sigma_{=} = \{x \leftarrow 0, z \leftarrow 0, u \leftarrow 0\}$ is $y = y$ and hence obviously not orientable.

Another instance of this equation is $x = -(-(x))$ reachable with the unifier $\sigma_{\rightarrow} = \{y \leftarrow 0,\ z \leftarrow 0,\ u \leftarrow 0\}$. This instance is essential for the completion as directed rule $-(-(x)) \rightarrow x$ and occurs only as instance of equations of the presented type when running AC1-completion.

This problem could be fixed by selecting appropriate instances of equations but that corresponds exactly to choosing suitable AC-unifiers. AC-completion through AC1-unification and disunification (see [BB89, KK89]) does not behave essentially better than AC-completion because the problem is not fixed to the number of unifiers, but inherent to the structure of the AC1-unifiers. For this method an additional problem is that the reduction operation must be restricted considerably when constraints are used (see [Mah91][(ii)]).

An additional unification algorithm included in the HADES-system is that for Abelian groups (AG). Without free function symbols this algorithm is unitary (like that for AC1) but with function symbols the number of unifiers explodes (comparatively to AC1).

Other experiments could be made using HADES' string (semigroup, associative, SG) unification algorithm also without promising results.

HADES additionally contains an algorithm for the unification in Boolean rings [CR89]. Boolean unification can for example be used in the field of hardware verification. W. Büttner and H. Simonis embedded it into a Prolog system for this purpose [BS87]. We did not make experiments with Boolean completion. But it is evident that the difficulties are the same as for AC1-unification because the theory is also unitary and the collapsing axioms constrain the ordering in a comparable manner.

For infinitary theories the same arguments have to be considered as for narrowing (see chapters 3 and 4). Special algorithms like that for associativity can be seen as specializations of this technique. One of the main problems is to control the unification algorithm such that it enumerates the unifiers in a heuristically useful order. A typical effect of taking the first solution first is, that it behaves worse than a really random choice (see [Gee92, section 4.2], for example). This random or a "best" choice cannot be done when having infinitary many solutions but can be approximated by computing "n" solutions and then choosing randomly or the best one, respectively.

The experiments with theory unification algorithms show that they are not very suitable in the context of theorem proving if they are used as they are. This is mainly due to the "uninformed" results they return. We think that it would be very promising to develop control mechanisms for theory unification algorithms, which enforce them to first return the "useful" results or constrain the results if they are used in special contexts.

[(ii)] Unfortunately the completion operation presented there is not correct. The constraints do not contain enough information to eliminate all AC1 critical pairs and hence too many equations can be inferred.

6.6 Towards a Human-Oriented Presentation of Equational Proofs

A theorem proving system is worthless if the proofs cannot be presented in a manner that human beings can understand them. Of course this is also the case for the MKRP protocols given in the appendix.

The proof transformation system GENTZEN was developed to transform a resolution proof given in form of a refutation graph [Sho76, Eis88] into a natural deduction proof [Jas33, Gen35]. Rules are incorporated to avoid the frequent use of proofs by contradiction and to select lemmata or perform case analysis. The system of transformation rules is described in detail in C. Lingenfelder's thesis [Lin90].

Together with C. Lingenfelder we extended the system in two directions: embedding equational reasoning into refutation graphs [LP91b] and constructing equation chain proofs from equality solution graphs [LP91a]. Equality solution graphs are special instances of the equation graphs introduced by K. Bläsius (see section 2.4.2), representing solutions for equality problems. They are the final states of equation graphs. In the transformation process of equational proofs they play the role of refutation graphs in the transformation of resolution proofs.

For the solution of the embedding problem two further rules in the natural deduction calculus and in the transformation system are necessary. In addition to G. Gentzen's original rules for his calculus $\mathcal{NK}$ we needed rules to handle the equality predicate. So we added the following rules:

Rule of Reflexivity (*Ref*): $\dfrac{}{\mathcal{A} \vdash t = t}$

Rule of Equality (=): $\dfrac{\mathcal{A} \vdash F[s] \quad \mathcal{B} \vdash s = t}{\mathcal{A}, \mathcal{B} \vdash F[t]}, \dfrac{\mathcal{A} \vdash F[t] \quad \mathcal{B} \vdash s = t}{\mathcal{A}, \mathcal{B} \vdash F[s]}$

In general C. Lingenfelder's method works by translating refutation graphs into Gentzen proofs. The equality applications can be represented in refutation graphs using special three literal clauses $P[t] \wedge t = s \Rightarrow P[s]$ simulating the replacement of t by s in $P[t]$. Hence it was necessary to add rules translating theses clauses into applications of the rules *Ref* and =. The $\mathcal{G}_{\mathcal{R}}$s in the following two rules represent refutation graphs. The $\mathcal{A}$s are assumptions of Gentzen proof lines.

The rule **E-=⊥** translates $s \neq s$ — $x = x$ resolutions into applications of *Ref*:

$$\left.\begin{array}{llll}(\gamma) & \mathcal{A} & \vdash \mathit{False} & \mathcal{G}_{\mathcal{R}}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{llll}(\alpha) & & \vdash s = s & \mathit{Ref} \\ (\beta) & \mathcal{A} & \vdash s \neq s & \mathcal{G}'_{\mathcal{R}} \\ (\gamma) & \mathcal{A} & \vdash \mathit{False} & \mathit{Contra}\end{array}\right.$$

E-= corresponds to the = rule:

$$\left.\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash s = t & \mathit{Rule}\ \mathcal{R} \\ (\gamma) & \mathcal{A} & \vdash F[t] & \mathcal{G}_{\mathcal{R}}\end{array}\right\} \rightsquigarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash s = t & \mathit{Rule}\ \mathcal{R} \\ (\beta) & \mathcal{A} & \vdash F[s] & \mathcal{G}'_{\mathcal{R}} \\ (\gamma) & \mathcal{A} & \vdash F[t] & = (\beta, \alpha)\end{array}\right.$$

For the second task dealing with purely equational proofs a new "equation chain proof" calculus with obvious rules (see the example below) is used as goal calculus. We started the translation process with equality graphs, for example: the one depicted in figure 2.1e, and the following rules for the translation into equation chains:

Insert:

$(\alpha) \quad \ldots s_1 = s_2 \ldots \quad (\ldots \mathcal{G}_{\mathcal{E}} \ldots)$

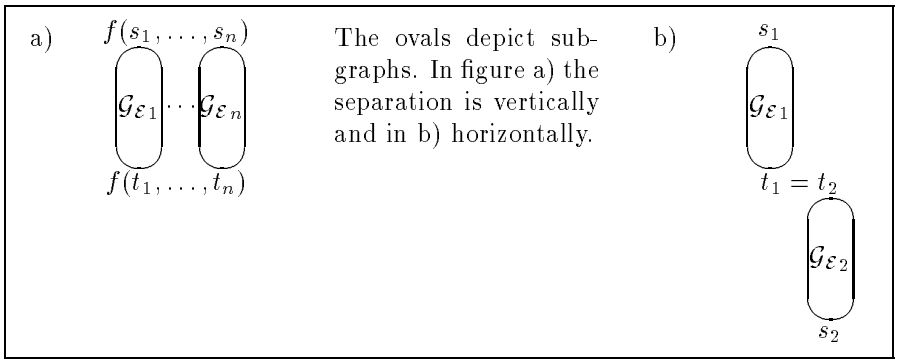
$\rightsquigarrow \quad (\alpha) \ldots s_1 = t_1 = t_2 = s_2 \ldots \quad (\ldots \mathcal{G}_{\mathcal{E}1}, t_1 = t_2, \mathcal{G}_{\mathcal{E}2} \ldots)$

if $\mathcal{G}_{\mathcal{E}}$ is an equality solution graph constructed corresponding to case 3 of the definition for equation solution graphs [LP91a, page 315] (see figure 6.5b) and the terms of the splitting equation $t_1 = t_2$ represent instances of the top level terms of an equality solution graph.

Decomposition:

$(\alpha) \quad \ldots f t_1 \ldots t_n = f s_1 \ldots s_n \ldots \quad (\ldots \mathcal{G}_{\mathcal{E}} \ldots)$

Figure 6.5: Splitting of Equation Solution Graphs



$\rightsquigarrow \quad (\alpha) \quad \ldots ft_1 \ldots t_n = fs_1 t_2 \ldots t_n = \ldots = fs_1 \ldots s_n \ldots \qquad (\ldots \mathcal{G}_{\mathcal{E}1}, \ldots, \mathcal{G}_{\mathcal{E}n} \ldots)$

if $\mathcal{G}_{\mathcal{E}}$ is an equality solution graph constructed corresponding to case 2 of the definition for equation solution graphs [LP91a, page 315] (see figure 6.5a).

Lemma:

$$(\beta) \quad \ldots t' = s' \ldots \quad (\ldots \mathcal{G}_{\mathcal{E}} \ldots) \qquad \rightsquigarrow \qquad \begin{cases} (\alpha) & t = s & (\mathcal{G}_{\mathcal{E}}) \\ (\beta) & \ldots t' = s' \ldots & (\ldots \alpha \ldots) \end{cases}$$

If $t' = s'$ is an instance of $t = s$, which is the pair of top level terms of $\mathcal{G}_{\mathcal{E}}$.

These rules fit perfectly well into C. Lingenfelder's original framework and all the features developed for his system can be easily adapted to equality proofs.

A proof for the simple example 2.16 (see also B.1.2) with the equality solution graph depicted in figure 2.1e can be developed with the second rule system beginning with the protocol of the MKRP-system, proceeding with the equality solution graph constructed from the protocol, and ending with the output:

$$\begin{array}{lcll} --a & = & --a + 0 & (x = x + 0) \\ & = & --a + (-a + a) & (0 = -x + x) \\ & = & (--a + -a) + a & (Associativity) \\ & = & 0 + a & (-x + x = 0) \\ & = & a & (0 + x = x) \end{array}$$

This seems to be the most natural representation for purely equational proofs.

Of course these mechanisms do not solve all problems with presentation of proofs. There is the need for a representation of narrowing steps and especially to protocol the intrinsics of theory unification algorithms.

Chapter 7

Conclusion

Our main conclusion is that a general equality reasoning procedure should include Knuth-Bendix completion in some way. Even if only a subset of equations can be directed it is still worthwhile to have the completion procedure around. Almost every interesting mathematical theory has a part consisting of directable unit equations, albeit it may not be possible for this subset to be "completed". The Knuth-Bendix algorithm is the most successful way to derive new interesting equations of which some are needed for almost every proof in the theory. The strong restrictions of equation application (directed and reduction) ensure that many interesting problems can be solved. In its constraining effect the usage of the Knuth-Bendix procedure is comparable to the Waltz-effect in contrast to the findings of K. Bläsius for his proof method [Blä86].[(i)]

The constraining effects equalize the usually negative forward reasoning character of the completion approach with its ignorance of goals. The forward search space becomes smaller than the corresponding backwardly directed one.

In addition other equality reasoning methods can be integrated together with the completion approach into one system. Narrowing can be used to obtain a proof style method retaining the advantages of E-resolution without inheriting the disadvantages. The integration of other inference rules as for example an induction step in an analogous form could be the goal of further research in this direction. The main objective must be to retain the equality reasoning power of the completion procedure whenever the equality predicate is present.

The superposition calculus with maximal literals resembles the Prolog SLD-strategy with head literal and indeed a similar compilation approach can be developed but unfortunately the gain of efficiency is not enough to accept the complications of the programs due to the compilation prerequisites.

One reason for this unsatisfactory outcome is that the forward reasoning nature of completion requires the dynamic construction of new programs. Another point is that a normal C-implementation is already so fast that compilation on the standard architecture does not enhance the efficiency to a reasonable extent. A third argument against a compilation approach for completion in general is the distribution of resources used by the single operations of the completion procedure. None of the operations is so dominant that compilation is valuable.

The extended selection of available heuristics and strategies gives evidence that a multi agent approach like that of J. Denzinger [Den91] can be extended to non-unit clauses.

Structure in axioms and theorems should be considered wherever this is possible. Unfortunately "normal" mathematical theories seem to have no structure usable for difference reduction in general, but at least this structure is not yet detected. This does not mean that knowledge should not be stored

[(i)] The Waltz-effect is exploited for constraint satisfaction in Vision to derive consistent possibilities in interpreting the topology of objects in a picture (see for example [Ric83b, pages 351-358]).

and exploited for finding proofs, and it does also not exclude to use structure for other purposes as for example induction.

We think that an equational calculus should fulfill three reduction properties: first it should reduce to Knuth-Bendix completion in the case of unit equations, which is ensured by almost all superposition strategies.

A second reduction is to classical narrowing in the case of equation solving. If the system can be completed to a canonical one this reduction can be achieved by interleaving the superposition approach with narrowing steps.

Thirdly it should reduce to Horn equational logic in the case of Horn clauses, which can be established for example by the two strategies of N. Dershowitz [Der91].

It is possible to integrate the main heuristics of the field with these three reduction properties using a superposition calculus as logical basis.

This thesis shows the adequacy of the superposition calculus for general automated theorem proving with equality. Furthermore it is possible to integrate almost all known techniques and heuristics of the field into this calculus. This approach is compatible with various calculus independent tools that were developed within our group and that are believed to be essential for a strong system, like the terminator and theory unification.

We conclude this work on integration with a general remark: whether something fits into your framework just depends on how you see it.

Appendix A

Sparc

This chapter contains the central part of the concrete realization of the ideas of chapter 5 for a Sun Sparc station. The following three Sparc assembler procedures realize patterns for the generation of terms, variables and term list cons cells. For explanations of the assembler see the SPARC architecture manual [Sun87].

A.1 Compound Term and Constant

```
_AT:    save    %sp,-128,%sp
        ba      TBegin
        nop
        nop                     Check slot.
        nop                     Binding.
        nop                     Type.
TBegin: ld      [%i0+36],%o0    Get other topsymbol from the input address in register i0. The first
                                19 bits are zeroed.
        sll     %o0,19,%o0
        srl     %o0,19,%o0
        cmp     %o0,111         Compare the own topsymbol with the other one.  111 is just an
                                arbitrary default.
        bne     TFalse          Clash.
        ld      [%i0+60],%o0    Compute the address of the other termlist to be input to the sub-
                                termlist function.
        sll     %o0,2,%o0       Compute the relative address.
        add     %o0,%i0,%o0     Compute the absolute address.
        add     %o0,60,%o0
        call    _ATl            Call the subtermlist function, for constants nop.
        nop
        cmp     %o0,0           Handle result of subtermlist call.
        be      TFalse
        mov     1,%i0           Move true into the result register.
        ba      TEnd
        nop
TFalse: mov     0,%i0           Move false into the result register.
TEnd:   ret
        restore
```

Such a term is generated by the C-function `TCreate`:

```
typedef unsigned int *termlist;

typedef unsigned int *term;

term TCreate (unsigned int top, unsigned int type, termlist tl)
{ term new;
  unsigned int i;
  if (type == 0)
  { new = VAlloc();
    new[VSetBindIndex] =
    CallLocate((unsigned int)((char *)ASetBind),
               (unsigned int)new+VSetBindRel);
    new[VCheckBindIndex] =
    CallLocate((unsigned int)((char *)ACheckBind),
               (unsigned int)new+VCheckBindRel); }
  else { new = TAlloc();
         TTopSet(new,top);
         TSubtermsSet(new,tl); }
  TTypeSet(new,type);
  CheckSlotSet(new,0);
  TBindingReset(new);
  return new; }
```

`VAlloc` and `TAlloc` copy the code from the master copies `VPattern` and `TPattern`.

```
term VAlloc()
{ void *new;
  new = OwnAlloc(VLength, "Variable");
  bcopy((char *)VPattern, new, VLength);
  return (term) new; }

term TAlloc()
{ void *new;
  new = OwnAlloc(TLength, "Term");
  bcopy((char *)TPattern, new, TLength);
  return (term) new; }
```

The patterns are connected to the mnemonic written assembler programs for terms and variables via the following declarations.

```
char * VPattern = (char *)AVar;

char * TPattern = (char *)AT;
```

A.2 Variable

```
_AVar:  save    %sp,-128,%sp
        ba      VBegin
        nop
        nop                     Check slot
        nop                     Binding
        nop                     Type
VBegin: mov     %i0,%o0         Give input argument to ACheckBind or ASetBind
        ba      VCB             Go to ACheckBind call (only for bound variables, otherwise nop)
```

```
        nop
        nop
        call    _ASetBind       Sets the binding and adjoins the variable to the bound variables
        nop
        ba      VEnd
        nop
        nop
        nop
VCB:    nop
        nop
        call    _ACheckBind
        nop
VEnd:   mov     %o0,%i0         Return what ACheckBind or ASetBind returns
        ret
        restore
```

Variables are also generated using the C-function of the previous section.

A.3 Term List Cons Cell

```
_ATl:   save    %sp,-128,%sp
        ld      [%i0+20],%o0 Get relative address of AT
        sll     %o0,2,%o0    Compute
        add     %o0,%i0,%o0  absolute
        add     %o0,20,%o0   address
        call    _AT          Call match for first element in the input list
        nop
        cmp     %o0,0
        be      TlFalse
        ld      [%i0+52],%o0 Get relative address of rest ATl
        sll     %o0,2,%o0    Compute
        add     %o0,%i0,%o0  absolute
        add     %o0,52,%o0   address
        call    _ATl
        nop
        cmp     %o0,0
        be      TlFalse
        mov     1,%i0        Match successful
        ba      TlEnd
        nop
TlFalse:mov     0,%i0        Match failed
TlEnd:  ret
        restore
```

Such calls are constructed using the C-function `TlCons`.

```
termlist TlCons(term t, termlist tl)
{ termlist new;
  unsigned int i;
  new = TlAlloc();
  TlFirstSet(new, t);
  TlRestSet(new, tl);
  return new; }
```

`TlAlloc` copies the assembler code from the master copy `TlPattern`.

```
termlist TlAlloc()
{ void *new;
  new = OwnAlloc(TlLength, "TlAlloc");
  bcopy((char *)TlPattern, new, TlLength);
  return (termlist) new; }
```

The connection is again done with a declaration:

```
char * TlPattern = (char *)ATl;
```

A.4 Bindings

Setting bindings is implemented by changing the jump commands in the variable data structure.

A.4.1 Setting Bindings

```
_ASetBind: save    %sp,-128,%sp
           st      %i0,[%i7-24]          Store binding.
           sethi   %hi(_AVar),%g1        Get call for AVar.
           add     %g1,%lo(_AVar),%g1
           ld      [%g1+28],%l0          Get VCB jump.
           st      %l0,[%i7-12]          Store jump.
           mov     %i7,%o0
           sub     %o0,40,%o0            Address of the actually bound variable (self).
           sethi   %hi(_bound),%g1       Get the stored list of all bound variables.
           ld      [%g1+%lo(_bound)],%o1
           call    _TlCons,0             Adjoin the actually bound variable itself to the list of
                                         all bound variables.
           nop
           sethi   %hi(_bound),%g1       Store the list of bound variables.
           st      %o0,[%g1+%lo(_bound)]
           mov     1,%i0
           ret
           restore
```

A.4.2 Checking Bindings

```
_ACheckBind: save    %sp,-128,%sp
             ld      [%i7-56],%o0 Get Binding
             mov     %i0,%o1
             call    _TEqual      Compare Bind with Input
             nop
             mov     %o0,%i0
             ret
             restore
```

Of course there are some special procedures written in the programming language C that deal with the above assembler code. Here we only gave the main declarations representing terms and termlists.

Appendix B

Commented Examples

The following examples are taken from arithmetic and algebra, others are just constructed to test certain aspects of theorem proving systems. The usage of arithmetic and algebraic examples should not propose that a general reasoning method is adequate to prove them, especially when algorithms exist to solve whole classes of problems. We only think that these well elaborated problems are more suitable for comparisons with other theorem provers and that it is more convenient to think about known concepts than strangely constructed ones. We focus on examples using the equality predicate because our goal was to enhance the power of a resolution theorem prover and leave the program work as before when no equations are present (see section 3.2.2).

E. Lusk and R. Overbeek [LO84] published a set of six equality problems without conditions that should be useful to check the power of an equality reasoning procedure (B.1.1, B.1.2, B.1.3, B.1.4, B.1.5, B.1.6). Examples B.1.1 and B.1.2 are trivial, examples B.1.3 through B.1.5 make some trouble but no problems, example B.1.6 could not be solved by the MKRP-system.

The next selection of examples is taken from F. Pelletier [Pel86] (B.2.1, B.2.2, B.2.3, B.2.4, B.2.5, B.2.6, B.2.7, B.2.8, B.2.9, B.2.10, B.2.11, B.2.12, B.2.13, B.2.14).

There are other well-known sets of examples, we cannot even cite all of them here, but we selected some simpler problems from various benchmarks in section B.3. L. Wos published examples in his book on the problems of automated theorem proving [Wos88]. L. Wos and W. McCune gave examples from combinator theory [WM88], which are classical challenging problems for pure equational reasoning (a simpler one of them is proved as example B.3.3). W. McCune selected various examples from lattice theory [McC88]; we could not prove any one of them. R. Stevens gave examples from the theory of non-associative rings [Ste88], where we also selected one for our tests but were not successful. Another interesting domain of pure equality reasoning is that of finitary groups (Z22 in section B.3.6, Z29 which can be seen as a challenge problem in example B.3.7). The hint to use the finite group examples came from J. Kalman. They are special string rewriting systems.

We focus on a wide range of more simpler examples than on real challenging ones. The only exception is the completion of the Z29 axioms (example B.3.7) for which we could generate a canonical rewrite system with the C-version of our system. The given examples are not selected for the purpose to evaluate the theorem prover as it is, but to compare equality reasoning methods in their ability to cooperate with an existing theorem prover and their resources in efficiency enhancement.

Now we come to the examples. The default reduction ordering for the system is a lexicographic recursive path ordering with the precedence `* > - > + > 1 > 0`. We changed it for some of the examples. The selected one is explained there. The number of paramodulation steps really performed is not depicted in the protocol and so we give it after the examples in a separate table.

The asterisks in the protocols label the axioms and derived clauses really used in the proof. The Markgraf-Karl system is a sorted theorem prover, but for the most equality reasoning examples no sorts

are used. This is documented in the protocol by the top sort `Any`. Positive literals are labeled with `+`, negative ones with `-`. For a detailed description of the language see [Wal82, Prä92b].

The clauses in the protocol are labeled with a unique number preceded by characters denoting their origin with the following meaning:

`A` Axiom
`D` Double literal removal
`F` Factor
`I` Instance
`P` Paramodulant
`R` Resolvent
`RS` Rewrite Symmetry
`RW` Rewrite
`T` Theorem

The numbers of the clauses in the protocol are not related to the numbers of clauses generated during the proof, they get their numbers in the protocol module due to some mystery. The computation times given in this thesis are mostly computed on a Symbolics UX1200.

B.1 Lusk/Overbeek

We begin this section with a short discussion of the results. Table B.1 shows a statistic of the necessary steps for the examples and we give the ratio *number of performed steps / number of proof steps* (g-penetrance) in a second column. Numbers smaller than one stem from the reduction steps which are not counted as paramodulation steps, because they are in some sense "deterministic". The third column gives the ratio when only completion steps in the proof are counted (this is the most interesting column). The table is another hint that the completion process can be seen as a very straightforward lemma generation. With a corresponding selection function and a lookahead that should be more efficient than ours, there are almost all produced clauses useful for the proof.

Table B.1: Ratio Useful Steps

Example	Steps	Ratio with rewrites	Ratio without rewrites
B.1.1	6	0.75	1.2
B.1.2	7	1	1.75
B.1.3	41	0.89	1.78
B.1.4	18	0.34	1.29
B.1.5	20	1.43	2.22
B.1.6	∞	∞	∞

Of course it is a little dishonest when we just count the generation of a new rule as one inference step. Another possibility would be to count the generation of critical pairs, that is, each resolution and paramodulation possibility as one inference step. Then we would have a considerably worse ratio. But the heuristically controlled selection justifies our counting.

B.1.1 Example 1

The first theorem states that every group with $x + x = 0$ is commutative, and this problem is very trivial using the completion technique. For a comparison with the decomposition approach based on universal unification see section 2.4.2.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
          A4:   All x:Any + =(+(-(x) x) 0)
        * A5:   All x:Any + =(+(x x) 0)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T6: - =(+(c_1 c_2) +(c_2 c_1))

Refutation:
===========

A5,1 & A2,1     --> * P1:    All x,y:Any + =(+(0 y) +(x +(x y)))
P1,1 & A3       --> * RW2:   All x,y:Any + =(y +(x +(x y)))
A5,1 & RW2,1    --> * P3:    All x:Any + =(x +(x 0))
A5,1 & A2,1     --> * P8:    All x,y:Any + =(0 +(y +(x +(y x))))
P8,1 & RW2,1    --> * P9:    All x,y:Any + =(+(y +(x y)) +(x 0))
P9,1 & P3       --> * RW10:  All x,y:Any + =(+(y +(x y)) x)
RW10,1 & RW2,1  --> * P13:   All x,y:Any + =(+(y x) +(x y))
P13,1 & T6,1    --> * R14:   []
```

B.1.2 Example 2

The second theorem states that the inverse of a group is an involution, and this problem is trivial too, because for a non-commutative group there exists a complete and confluent rewrite system, such that in this theory all purely equational theorems can be solved.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T5: - =(-(-(c_1)) c_1)

Refutation:
===========

A4,1 & A2,1     --> * P1:    All x,y:Any + =(+(0 y) +(-(x) +(x y)))
P1,1 & A3       --> * RW2:   All x,y:Any + =(y +(-(x) +(x y)))
A4,1 & RW2,1    --> * P4:    All x:Any + =(x +(-(-(x)) 0))
P4,1 & RW2,1    --> * P12:   All x:Any + =(0 +(-(-(-(x))) x))
P12,1 & RW2,1   --> * P13:   All x:Any + =(x +(-(-(-(-(x)))) 0))
P13,1 & P4      --> * RW14:  All x:Any + =(x -(-(x)))
RW14,1 & T5,1   --> * R15:   []
```

B.1.3 Example 3

The third example comes from ring theory and is therefore more complicated but not really difficult because the commutativity of addition is not involved and so no undirectable equations must be applied.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)
          A5:   All x,y:Any + =(+(y x) +(x y))
          A6:   All x,y,z:Any + =(*(*(z y) x) *(z *(y x)))
        * A7:   All x,y,z:Any + =(*(+(z y) x) +(*(z x) *(y x)))
        * A8:   All x,y,z:Any + =(*(z +(y x)) +(*(z y) *(z x)))
        * A9:   All x:Any + =(*(x x) x)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T10: - =(*(c_1 c_2) *(c_2 c_1))

Refutation:
===========

A4,1 & A2,1         --> * P1:     All x,y:Any + =(+(0 y)
                                                  +(-(x) +(x y)))
P1,1 & A3           --> * RW2:    All x,y:Any + =(y +(-(x) +(x y)))
A3,1 & RW2,1        --> * P3:     All x:Any + =(x +(-(0) x))
A4,1 & RW2,1        --> * P4:     All x:Any + =(x +(-(-(x)) 0))
P3,1 & RW2,1        --> * P5:     All x:Any + =(x +(-(-(0)) x))
P5,1 & A4,1         --> * P6:     + =(-(0) 0)
P4,1 & RW2,1        --> * P13:    All x:Any + =(0 +(-(-(-(x))) x))
P13,1 & RW2,1       --> * P14:    All x:Any + =(x +(-(-(-(-(x)))) 0))
P14,1 & P4          --> * RW15:   All x:Any + =(x -(-(x)))
P4,1 & RW15         --> * RW16:   All x:Any + =(x +(x 0))
RW15,1 & A4,1       --> * P19:    All x:Any + =(+(x -(x)) 0)
P19,1 & A2,1        --> * P21:    All x,y:Any + =(0 +(y +(x -(+(y x)))))
P21,1 & RW2,1       --> * P22:    All x,y:Any + =(+(y -(+(x y)))
                                                  +(-(x) 0))
P22,1 & RW16        --> * RW23:   All x,y:Any + =(+(y -(+(x y))) -(x))
RW23,1 & RW23,1     --> * P26:    All x,y:Any + =(+(-(+(y x)) -(-(y)))
                                                  -(x))
P26,1 & RW15        --> * RW27:   All x,y:Any + =(+(-(+(y x)) y) -(x))
A3,1 & A7,1         --> * P28:    All x,y:Any + =(*(y x) +(*(0 x) *(y x)))
P28,1 & RW23,1      --> * P29:    All x,y:Any + =(+(*(y x) -(*(y x)))
                                                  -(*(0 x)))
P29,1 & P19         --> * RW30:   All x:Any + =(0 -(*(0 x)))
RW30,1 & A4,1       --> * P31:    All x:Any + =(+(0 *(0 x)) 0)
P31,1 & A3          --> * RW32:   All x:Any + =(*(0 x) 0)
A4,1 & A7,1         --> * P37:    All x,y:Any + =(*(0 y)
                                                  +(*(-(x) y) *(x y)))
P37,1 & RW32        --> * RW38:   All x,y:Any + =(0 +(*(-(y) x) *(y x)))
A9,1 & RW38,1       --> * P39:    All x:Any + =(0 +(*(-(x) x) x))
P39,1 & RW27,1      --> * P42:    All x:Any + =(+(-(0) *(-(x) x)) -(x))
P42,1 & P6          --> * RW43:   All x:Any + =(+(0 *(-(x) x)) -(x))
RW43,1 & A3         --> * RW44:   All x:Any + =(*(-(x) x) -(x))
RW15,1 & RW44,1     --> * P49:    All x:Any + =(*(x -(x)) -(-(x)))
P49,1 & RW15        --> * RW50:   All x:Any + =(*(x -(x)) x)
A3,1 & A8,1         --> * P64:    All x,y:Any + =(*(y x) +(*(y 0) *(y x)))
P64,1 & RW23,1      --> * P65:    All x,y:Any + =(+(*(y x) -(*(y x)))
                                                  -(*(y 0)))
P65,1 & P19         --> * RW66:   All x:Any + =(0 -(*(x 0)))
RW66,1 & A4,1       --> * P67:    All x:Any + =(+(0 *(x 0)) 0)
```

```
P67,1 & A3          --> * RW68:   All x:Any + =(*(x 0) 0)
A4,1 & A8,1         --> * P73:    All x,y:Any + =(*(y 0)
                                                 +(*(y -(x)) *(y x)))
P73,1 & RW68        --> * RW74:   All x,y:Any + =(0 +(*(y -(x)) *(y x)))
RW50,1 & RW74,1     --> * P75:    All x:Any + =(0 +(x *(x x)))
P75,1 & A9          --> * RW76:   All x:Any + =(0 +(x x))
RW76,1 & RW2,1      --> * P77:    All x:Any + =(x +(-(x) 0))
P77,1 & RW16        --> * RW78:   All x:Any + =(x -(x))
RW27,1 & RW78       --> * RW89:   All x,y:Any + =(+(+(y x) y) -(x))
RW89,1 & RW78       --> * RW90:   All x,y:Any + =(+(+(y x) y) x)
RW90,1 & A2         --> * RW91:   All x,y:Any + =(+(y +(x y)) x)
RW91,1 & A2,1
  --> * P109:   All x,y,z:Any + =(z +(y +(x +(z +(y x)))))
RW91,1 & P109,1
  --> * P114:   All x,y,z:Any + =(z +(y +(+(x y) +(z x))))
P114,1 & A2
  --> * RW115:  All x,y,z:Any + =(z +(y +(x +(y +(z x)))))
A9,1 & A7,1
  --> * P125:   All x,y:Any + =(+(y x) +(*(y +(y x)) *(x +(y x))))
P125,1 & A8
  --> * RW126:  All x,y:Any + =(+(y x) +(+(*(y y) *(y x)) *(x +(y x))))
RW126,1 & A9
  --> * RW127:  All x,y:Any + =(+(y x) +(+(y *(y x)) *(x +(y x))))
RW127,1 & A8
  --> * RW128:  All x,y:Any + =(+(y x) +(+(y *(y x)) +(*(x y) *(x x))))
RW128,1 & A9
  --> * RW129:  All x,y:Any + =(+(y x) +(+(y *(y x)) +(*(x y) x)))
RW129,1 & A2
  --> * RW130:  All x,y:Any + =(+(y x) +(y +(*(y x) +(*(x y) x))))
RW130,1 & RW115,1
  --> * P131:   All x,y:Any + =(*(y x) +(y +(+(*(x y) x) +(y x))))
P131,1 & A2
  --> * RW132:  All x,y:Any + =(*(y x) +(y +(*(x y) +(x +(y x)))))
RW132,1 & RW91     --> * RW133:  All x,y:Any + =(*(y x) +(y +(*(x y) y)))
RW133,1 & RW91     --> * RW134:  All x,y:Any + =(*(y x) *(x y))
RW134,1 & T10,1    --> * R135:   []
```

B.1.4 Example 4

The fourth example is another special theorem of group theory and in complexity comparable to the third one. Again no unorientable equation is necessary and therefore no real problem arises. Here an additional feature of the Markgraf-Karl system comes into the game. It detects definitory equations and replaces the definiens at every occurrence by the definiendum. This is sometimes very useful because it reduces the clause set and especially the number of function symbols. Here `A6` is taken as definition for `comm` and so `comm` is completely eliminated by preprocessing operations. For this example we chose the usual Knuth-Bendix ordering for the completion of groups [KB70] such that clause `RW21` is directed from left to right. The precedence is `- > + > 1`, and the weights are `+: 1, -: 0, 0: 0`. In this way the refutation is found faster, because the search space is smaller.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)
        * A5:   All x:Any + =(+(+(x x) x) 0)
        * A6:   All x,y:Any + =(comm(y x) +(y +(x +(-(y) -(x)))))

Initial Operations on Axioms
============================
```

```
A5,1 & A2 --> * RW1: All x:Any + =(+(x +(x x)) 0)

Set of Theorem Clauses Resulting from Normalization
===================================================

       * T7: - =(comm(comm(c_1 c_2) c_2) 0)

Initial Operations on Theorems
==============================

T7,1 & A6  --> * RW2:
  - =(+(comm(c_1 c_2) +(c_2 +(-(comm(c_1 c_2)) -(c_2)))) 0)
RW2,1 & A6 --> * RW3:
  - =(+(+(c_1 +(c_2 +(-(c_1) -(c_2))))
        +(c_2 +(-(+(c_1 +(c_2 +(-(c_1) -(c_2))))) -(c_2))))
      0)
RW3,1 & A2 --> * RW4:
  - =(+(c_1 +(+(c_2 +(-(c_1) -(c_2)))
              +(c_2 +(-(+(c_1 +(c_2 +(-(c_1) -(c_2))))) -(c_2)))))
      0)
RW4,1 & A2 --> * RW5:
  - =(+(c_1 +(c_2 +(+(-(c_1) -(c_2))
                   +(c_2 +(-(+(c_1 +(c_2 +(-(c_1) -(c_2))))) -(c_2))))))
      0)
RW5,1 & A2 --> * RW6:
  - =(+(c_1 +(c_2 +(-(c_1) +(-(c_2) +(c_2 +(-(+(c_1 +(c_2 +(-(c_1)
                                                          -(c_2)))))
                                             -(c_2)))))))
      0)

Refutation:
===========

A4,1 & A2,1    --> * P7:    All x,y:Any + =(+(0 y) +(-(x) +(x y)))
P7,1 & A3      --> * RW8:   All x,y:Any + =(y +(-(x) +(x y)))
RW6,1 & RW8    --> * RW9:
  - =(+(c_1 +(c_2 +(-(c_1) +(-(+(c_1 +(c_2 +(-(c_1) -(c_2))))) -(c_2)))))
      0)
A4,1 & RW8,1   --> * P14:   All x:Any + =(x +(-(-(x)) 0))
RW1,1 & RW8,1  --> * P15:   All x:Any + =(+(x x) +(-(x) 0))
RW1,1 & A2,1   --> * P16:   All x,y:Any + =(+(0 y) +(x +(+(x x) y)))
P16,1 & A2     --> * RW17:  All x,y:Any + =(+(0 y) +(x +(x +(x y))))
RW17,1 & A3    --> * RW18:  All x,y:Any + =(y +(x +(x +(x y))))
RW1,1 & RW18,1 --> * P19:   All x:Any + =(x +(x 0))
P14,1 & P19    --> * RW20:  All x:Any + =(x -(-(x)))
P15,1 & P19    --> * RW21:  All x:Any + =(+(x x) -(x))
RW1,1 & RW21   --> * RW22:  All x:Any + =(+(x -(x)) 0)
RW22,1 & A2,1  --> * P27:   All x,y:Any + =(+(0 y) +(x +(-(x) y)))
P27,1 & A3     --> * RW28:  All x,y:Any + =(y +(x +(-(x) y)))
A2,1 & RW22,1  --> * P32:   All x,y:Any + =(+(y +(x -(+(y x)))) 0)
P32,1 & RW8,1  --> * P33:   All x,y:Any + =(+(y -(+(x y))) +(-(x) 0))
P33,1 & P19    --> * RW34:  All x,y:Any + =(+(y -(+(x y))) -(x))
RW21,1 & A2,1  --> * P36:   All x,y:Any + =(+(-(y) x) +(y +(y x)))
RW34,1 & RW8,1 --> * P38:   All x,y:Any + =(-(+(y x)) +(-(x) -(y)))
RW9,1 & P38    --> * RW40:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(-(+(c_2 +(-(c_1) -(c_2)))) -(c_1))
                             -(c_2)))))
      0)
RW40,1 & P38   --> * RW41:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(+(-(+(-(c_1) -(c_2))) -(c_2)) -(c_1))
                             -(c_2)))))
      0)
RW41,1 & P38   --> * RW42:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(+(+(-(-(c_2)) -(-(c_1))) -(c_2)) -(c_1))
                             -(c_2)))))
      0)
```

```
RW42,1 & RW20  --> * RW43:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(+(+(-(-(c_2)) c_1) -(c_2)) -(c_1))
                             -(c_2)))))
     0)
RW43,1 & RW20  --> * RW44:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(+(+(c_2 c_1) -(c_2)) -(c_1)) -(c_2))))) 0)
RW44,1 & A2    --> * RW45:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(+(c_2 +(c_1 -(c_2))) -(c_1)) -(c_2))))) 0)
RW45,1 & A2    --> * RW46:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(c_2 +(+(c_1 -(c_2)) -(c_1))) -(c_2))))) 0)
RW46,1 & A2    --> * RW47:
  - =(+(c_1 +(c_2 +(-(c_1) +(+(c_2 +(c_1 +(-(c_2) -(c_1)))) -(c_2))))) 0)
RW47,1 & A2    --> * RW48:
  - =(+(c_1 +(c_2 +(-(c_1) +(c_2 +(+(c_1 +(-(c_2) -(c_1))) -(c_2)))))) 0)
RW48,1 & A2    --> * RW49:
  - =(+(c_1 +(c_2 +(-(c_1) +(c_2 +(c_1 +(+(-(c_2) -(c_1)) -(c_2))))))) 0)
RW49,1 & A2    --> * RW50:
  - =(+(c_1 +(c_2 +(-(c_1) +(c_2 +(c_1 +(-(c_2) +(-(c_1) -(c_2)))))))) 0)
A2,1 & RW21,1  --> * P51:   All x,y:Any + =(+(y +(x +(y x))) -(+(y x)))
P51,1 & P38    --> * RW52:  All x,y:Any + =(+(y +(x +(y x)))
                                           +(-(x) -(y)))
RW52,1 & P36,1 --> * P56:   All x,y:Any + =(+(-(y) +(x +(y x)))
                                           +(y +(-(x) -(y))))
RW20,1 & P56,1 --> * P57:   All x,y:Any + =(+(y +(x +(-(y) x)))
                                           +(-(y) +(-(x) -(-(y)))))
P57,1 & RW20   --> * RW58:  All x,y:Any + =(+(y +(x +(-(y) x)))
                                           +(-(y) +(-(x) y)))
RW50,1 & RW58  --> * RW59:
  - =(+(c_1 +(c_2 +(-(c_1) +(c_2 +(-(c_1) +(-(-(c_2)) c_1)))))) 0)
RW59,1 & RW20  --> * RW60:
  - =(+(c_1 +(c_2 +(-(c_1) +(c_2 +(-(c_1) +(c_2 c_1)))))) 0)
P36,1 & A2,1   --> * P61:   All x,y,z:Any + =(+(-(+(z y)) x)
                                             +(z +(y +(+(z y) x))))
P61,1 & A2     --> * RW62:  All x,y,z:Any + =(+(-(+(z y)) x)
                                             +(z +(y +(z +(y x)))))
RW62,1 & P38   --> * RW63:  All x,y,z:Any + =(+(+(-(z) -(y)) x)
                                             +(y +(z +(y +(z x)))))
RW63,1 & A2    --> * RW64:  All x,y,z:Any + =(+(-(z) +(-(y) x))
                                             +(y +(z +(y +(z x)))))
RW60,1 & RW64  --> * RW65:  - =(+(c_1 +(c_2 +(-(c_2) +(-(-(c_1)) c_1))))
                               0)
RW65,1 & RW20  --> * RW66:  - =(+(c_1 +(c_2 +(-(c_2) +(c_1 c_1)))) 0)
RW66,1 & RW21  --> * RW67:  - =(+(c_1 +(c_2 +(-(c_2) -(c_1)))) 0)
RW67,1 & RW28  --> * RW68:  - =(+(c_1 -(c_1)) 0)
RW68,1 & RW22  --> * RW69:  - =(0 0)
RW69,1 & A1,1  --> * R70:   []
```

B.1.5 Example 5

The fifth example is taken from the less known theory of ternary algebra and we have to say a few words about the problem and our proof. In most cases the theory is given with an additional axiom: a right inverse `*(x y -(y)) = x`. But only one of the inverses is necessary. Normally a Knuth-Bendix reduction ordering, which we used for this example, sets the parentheses right-associative so that for example a(b(c(d e))) is the normal form and not (((a b)c)d)e. In this case omitting the left inverse causes no difficulties and the theorem can be proven despite of the divergence of the completion algorithm. But omitting the right inverse induces trouble, the left side of the inverse rule cannot be unified with the left side of the equation `P20`. So the other way of setting the parentheses must be chosen to find a proof with this method. It is a commonly used technique to define orderings from left to right or right to left for different operators in term rewriting. Using the left to right ordering and unfailing completion such that the necessary derived unorientable equations can be applied in both directions, increases the

search space enormously such that MKRP does not find the solution, but it is not impossible to find it as shown by J. Denzinger [Den91].

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z,u,v:Any + =(*(*(v u z) y *(v u x))
                                      *(v u *(z y x)))
        * A3:   All x,y:Any + =(*(y x x) x)
        * A4:   All x,y:Any + =(*(y y x) y)
        * A5:   All x,y:Any + =(*(-(y) y x) x)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T6: - =(*(c_1 -(c_1) c_2) c_2)

Refutation:
===========

A4,1 & A2,1     -->  * P1:    All x,y,z,u:Any + =(*(u z *(u u y))
                                                   *(u u *(x z y)))
P1,1 & A4       -->  * RW2:   All x,y,z:Any + =(*(z y *(z z x)) z)
RW2,1 & A4      -->  * RW3:   All x,y:Any + =(*(y x y) y)
A3,1 & A2,1     -->  * P7:    All x,y,z,u:Any + =(*(*(u z y) x z)
                                                   *(u z *(y x z)))
A5,1 & P7,1     -->  * P8:    All x,y,z:Any + =(*(*(z y -(x)) x y)
                                                 *(z y y))
P8,1 & A3       -->  * RW9:   All x,y,z:Any + =(*(*(z y -(x)) x y) y)
A4,1 & P7,1     -->  * P10:   All x,y,z:Any + =(*(*(z y x) x y) *(z y x))
P10,1 & RW9,1   -->  * P11:   All x,y,z:Any + =(*(*(z -(y) x) y x) x)
RW3,1 & A2,1    -->  * P20:   All x,y,z,u:Any + =(*(*(u z y) x u)
                                                   *(u z *(y x u)))
A4,1 & P20,1    -->  * P23:   All x,y,z:Any + =(*(*(z y x) x z) *(z y x))
P23,1 & P10,1   -->  * P25:   All x,y,z:Any + =(*(*(z y x) z x)
                                                 *(*(z y x) x z))
P25,1 & P23     -->  * RW26:  All x,y,z:Any + =(*(*(z y x) z x) *(z y x))
P11,1 & RW26,1  -->  * P27:   All x,y:Any + =(y *(x -(x) y))
P27,1 & T6,1    -->  * R28:   []
```

B.1.6 Example 6

E. Lusk and R. Overbeek introduced this problem to the community, which is probably the best known one in equality reasoning: "every ring with $x^3 = x$ is commutative." Many authors as for example M. Stickel [Sti84] and D. Kapur [KZ89] focused on it. They used special techniques to solve it and other related problems in the family $x^n = x$, especially they used completion modulo AC-unification. D. Kapur developed a special algorithm to handle these problems very efficiently. We used a completion procedure based on AC1-unification with constraints like this of C. and H. Kirchner [KK89], but we did not succeed in solving the problem with this technique (see section 6.5). The problem also was solved by J. Denzinger using a multi agent approach with parallel unfailing completion in about 700 seconds [Den91].

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
          A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
          A3:   All x:Any + =(+(0 x) x)
          A4:   All x:Any + =(+(-(x) x) 0)
          A5:   All x,y:Any + =(+(y x) +(x y))
          A6:   All x,y,z:Any + =(*(*(z y) x) *(z *(y x)))
```

```
          A7:   All x,y,z:Any + =(*(+(z y) x) +(*(z x) *(y x)))
          A8:   All x,y,z:Any + =(*(z +(y x)) +(*(z y) *(z x)))
          A9:   All x:Any + =(*(x *(x x)) x)


Set of Theorem Clauses Resulting from Normalization
===================================================

          T10: - =(*(c_1 c_2) *(c_2 c_1))
```

B.2 Pelletier

F. Pelletier published a graduated set of 75 problems for testing automatic theorem provers. We picked from this set all clearly specified problems dealing with the equality predicate. It must be noted that F. Pelletier's task 73 is the equality representation of the pigeon hole problem, which is in fact an infinitary set of problems. We just computed the 3-hole problem (B.2.14).

In table 6.1 we collected the statistics for F. Pelletier's examples. The last column gives the additional features of MKRP needed to find a proof.

B.2.1 Example 48

The most astonishing thing with the trivial example 48 is that it needs nine steps in the proof. It is one of the simplest possible examples with conditional equations.

```
Formulae Given to the Editor
============================

Axioms:   A = B OR C = D
          A = C OR B = D

Theorems: A = D OR B = C

Set of Axiom Clauses Resulting from Normalization
=================================================

            A1:   All x:Any + =(x x)
          * A2:   + =(a b)  + =(c d)
          * A3:   + =(a c)  + =(b d)

Set of Theorem Clauses Resulting from Normalization
===================================================

          * T4: - =(a d)
          * T5: - =(b c)

Refutation:
===========
A3,2 & A2,1    -->  * P1:   + =(a d)  + =(a c)  + =(c d)
P1,1 & T4,1    -->  * R2:   + =(a c)  + =(c d)
A3,2 & T5,1    -->  * P3:   - =(d c)  + =(a c)
P3,1 & R2,2    -->  * R4:   + =(a c)  + =(a c)
R4 1=2         -->  * D5:   + =(a c)
A2,1 & D5      -->  * RW6:  + =(c b)  + =(c d)
T4,1 & D5      -->  * RW7:  - =(c d)
RW6,2 & RW7,1  -->  * R8:   + =(c b)
R8,1 & T5,1    -->  * R9:   []
```

B.2.2 Example 49

Problem 49 combines conditional equations with other predicates. It states that if there are two objects such that all other objects are equal to one of them and if we have two constants with the property P then P holds universally.

```
Formulae Given to the Editor
============================

Axioms:   EX X,Y (ALL Z Z = X OR Z = Y)
          P(A) AND P(B)
          NOT A = B

Theorems: ALL X P(X)

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   + P(a)
        * A3:   + P(b)
        * A4:   - =(a b)
        * A5:   All x:Any + =(x c_2)  + =(x c_1)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T6: - P(c_3)

Refutation:
===========

A5,1 & A2,1     --> * P1:    + P(c_1)  + =(a c_2)
A5,1 & A3,1     --> * P3:    + P(c_1)  + =(b c_2)
P1,2 & A4,1     --> * P4:    - =(c_2 b)  + P(c_1)
P4,1 & P3,2     --> * R5:    + P(c_1)  + P(c_1)
R5 1=2          --> * D6:    + P(c_1)
A5,1 & A4,1     --> * P7:    - =(c_1 b)  + =(a c_2)
P7,2 & A2,1     --> * P8:    + P(c_2)  - =(c_1 b)
A5,1 & P8,2     --> * P9:    - =(c_1 c_1)  + =(b c_2)  + P(c_2)
P9,1 & A1,1     --> * R10:   + =(b c_2)  + P(c_2)
R10,1 & A3,1    --> * P12:   + P(c_2)  + P(c_2)
P12 2=1         --> * D13:   + P(c_2)
A5,1 & T6,1     --> * P23:   - P(c_1)  + =(c_3 c_2)
P23,1 & D6,1    --> * R24:   + =(c_3 c_2)
D13,1 & R24     --> * RW27:  + P(c_3)
RW27,1 & T6,1   --> * R32:   []
```

B.2.3 Example 51

Problems B.2.3, B.2.4, and B.2.5 are comparable to Andrews' challenging example for resolution theorem proving. The theorems contain equivalences which explode during the normalization. Such examples show that methods using clause normal form are often not adequate (see [Sie92, KP92], and also section 6.4).

```
Formulae given to the editor
============================

Axioms:   EX Z,W (ALL X,Y P(X Y) EQV (X = Z) AND Y = W)

Theorems: EX Z (ALL X (EX W (ALL Y P(X Y) EQV Y = W)) EQV X = Z)
```

```
Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x,y:Any - P(y x)  + =(y c_2)
        * A3:   All x,y:Any - P(y x)  + =(x c_1)
        * A4:   All x,y:Any + P(y x)  - =(y c_2)  - =(x c_1)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T5: All x,y:Any - P(f_3(y) f_1(x y))
                          - =(f_1(x y) x)  - =(f_3(y) y)
        * T6: All x,y:Any + P(f_3(y) f_1(x y))
                          + =(f_1(x y) x)  - =(f_3(y) y)
        * T7: All x,y:Any + P(f_3(y) x)  - =(x f_2(y))  + =(f_3(y) y)
          T8: All x,y:Any - P(f_3(y) x)  + =(x f_2(y))  + =(f_3(y) y)

Refutation:
===========

T7,1 & A2,1   --> * R1:    All x,y:Any - =(y f_3(x))  + =(f_1(x) x)
                                       + =(f_1(x) c_1)
R1,1 & A1,1   --> * R2:    All x:Any + =(f_1(x) x)  + =(f_1(x) c_1)
R2 (factor)   --> * F3:    + =(f_1(c_1) c_1)
F3,1 & T6,1   --> * P9:    All x:Any + P(c_1 f_2(x c_1))  + =(f_2(x c_1) x)
                                     - =(f_1(c_1) c_1)
P9,3 & F3     --> * RW10:  All x:Any + P(c_1 f_2(x c_1))  + =(f_2(x c_1) x)
                                     - =(c_1 c_1)
RW10,3 & A1,1 --> * R11:   All x:Any + P(c_1 f_2(x c_1))  + =(f_2(x c_1) x)
R11,1 & A3,1  --> * R12:   All x:Any + =(f_2(x c_1) x)  + =(f_2(x c_1) c_2)
R12 (factor)  --> * F13:   + =(f_2(c_2 c_1) c_2)
F13,1 & T5,1  --> * P15:   - P(f_1(c_1) c_2)  - =(f_2(c_2 c_1) c_2)
                           - =(f_1(c_1) c_1)
P15,3 & F3    --> * RW16:  - P(f_1(c_1) c_2)  - =(f_2(c_2 c_1) c_2)
                           - =(c_1 c_1)
RW16,2 & F13  --> * RW17:  - P(f_1(c_1) c_2)  - =(c_2 c_2)  - =(c_1 c_1)
RW17,1 & F3   --> * RW18:  - P(c_1 c_2)  - =(c_2 c_2)  - =(c_1 c_1)
RW18,2 & A1,1 --> * R19:   - P(c_1 c_2)  - =(c_1 c_1)
R19,2 & A1,1  --> * R20:   - P(c_1 c_2)
R20,1 & A4,1  --> * R21:   - =(c_1 c_1)  - =(c_2 c_2)
R21,1 & A1,1  --> * R22:   - =(c_2 c_2)
R22,1 & A1,1  --> * R23:   []
```

B.2.4 Example 52

Problem 52 is a symmetric version of B.2.3 and therefore we omit the proof. The produced proof differs from that of problem 51 because the automatically generated ordering for the Skolem functions led to a different operator ordering and in this case to a longer proof.

```
Formulae Given to the Editor
============================

Axioms:   EX Z,W (ALL X,Y P(X Y) EQV (X = Z AND Y = W))

Theorems: EX W (ALL Y (EX Z (ALL X P(X Y) EQV X = Z)) EQV Y = W)
```

B.2.5 Example 53

Example 53 is the most difficult of the equality problems given by F. Pelletier. The proof is done in two parts for the two directions of the equivalence. These parts are completely symmetric and the proofs are equal modulo the names of Skolem constants and functions, hence we omit the second splitpart. The symmetry contrasts to other examples where asymmetric effects occur due to the directability of

associativity or other asymmetric axioms. F. Pelletier formulated the problem not to be proven but to test the normalization operation of theorem provers. We think that this means that strong logical preprocessing operations during normalization are able to solve the problem. However, those of the MKRP-system were not able to do so.

Our option setting includes an option to use all possible instances of clauses if a finite domain is specified, in this case by the clause $x = c_2 \vee x = c_1$. In the given example this strategy is complete because the unique possibility to produce deeper terms is to paramodulate into variables, which is not necessary.

```
Formulae Given to the Editor
============================

Axioms:   EX X,Y NOT Y = X AND (ALL Z Z = X OR Z = Y)

Theorems: (EX Z (ALL X ((EX W (ALL Y (P (X Y) EQV Y = W))) EQV X = Z)))
          EQV (EX Z (ALL X ((EX W (ALL Y (P (Y X) EQV Y = W))) EQV X = Z)))

Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   - =(c_2 c_1)
        * A3:   All x:Any + =(x c_1)  + =(x c_2)

Set of Theorem Clauses Resulting from Normalization and Splitting
=================================================================

Splitpart 1

        * T4: All x,y:Any - P(f_1(y) f_2(x y))  - =(f_2(x y) x)  - =(f_1(y) y)
        * T5: All x,y:Any + P(f_1(y) f_2(x y))  + =(f_2(x y) x)  - =(f_1(y) y)
        * T6: All x,y:Any + P(f_1(y) x)  - =(x f_3(y))  + =(f_1(y) y)
        * T7: All x,y:Any - P(f_1(y) x)  + =(x f_3(y))  + =(f_1(y) y)
        * T8: All x,y:Any + P(y x)  - =(y f_4(x))  - =(x c_3)
        * T9: All x,y:Any - P(y x)  + =(y f_4(x))  - =(x c_3)
        * T10: All x,y:Any - P(f_5(y x) x)  - =(f_5(y x) y)  + =(x c_3)
        * T11: All x,y:Any + P(f_5(y x) x)  + =(f_5(y x) y)  + =(x c_3)

Splitpart 2    analogous to splitpart 1

Initial Operations on Theorems
==============================

Splitpart 1

T4 (instance)  --> * I1:  - P(f_1(c_2) f_2(c_2 c_2))
                          - =(f_2(c_2 c_2) c_2)  - =(f_1(c_2) c_2)
T4 (instance)  --> * I2:  - P(f_1(c_2) f_2(c_1 c_2))
                          - =(f_2(c_1 c_2) c_1)  - =(f_1(c_2) c_2)
T4 (instance)  --> * I3:  - P(f_1(c_1) f_2(c_2 c_1))
                          - =(f_2(c_2 c_1) c_2)  - =(f_1(c_1) c_1)
T4 (instance)  --> * I4:  - P(f_1(c_1) f_2(c_1 c_1))
                          - =(f_2(c_1 c_1) c_1)  - =(f_1(c_1) c_1)
T5 (instance)  --> * I5:  + P(f_1(c_2) f_2(c_2 c_2))
                          + =(f_2(c_2 c_2) c_2)  - =(f_1(c_2) c_2)
T5 (instance)  --> * I6:  + P(f_1(c_2) f_2(c_1 c_2))
                          + =(f_2(c_1 c_2) c_1)  - =(f_1(c_2) c_2)
T5 (instance)  --> * I7:  + P(f_1(c_1) f_2(c_2 c_1))
                          + =(f_2(c_2 c_1) c_2)  - =(f_1(c_1) c_1)
T5 (instance)  --> * I8:  + P(f_1(c_1) f_2(c_1 c_1))
                          + =(f_2(c_1 c_1) c_1)  - =(f_1(c_1) c_1)
T6 (instance)  --> * I9:  + P(f_1(c_2) c_2)  - =(c_2 f_3(c_2))
                          + =(f_1(c_2) c_2)
T6 (instance)  --> * I10: + P(f_1(c_2) c_1)  - =(c_1 f_3(c_2))
                          + =(f_1(c_2) c_2)
```

```
T6 (instance)  --> * I11: + P(f_1(c_1) c_2)  - =(c_2 f_3(c_1))
                          + =(f_1(c_1) c_1)
T6 (instance)  --> * I12: + P(f_1(c_1) c_1)  - =(c_1 f_3(c_1))
                          + =(f_1(c_1) c_1)
T7 (instance)  --> * I13: - P(f_1(c_2) c_2)  + =(c_2 f_3(c_2))
                          + =(f_1(c_2) c_2)
T7 (instance)  --> * I14: - P(f_1(c_2) c_1)  + =(c_1 f_3(c_2))
                          + =(f_1(c_2) c_2)
T7 (instance)  --> * I15: - P(f_1(c_1) c_2)  + =(c_2 f_3(c_1))
                          + =(f_1(c_1) c_1)
T7 (instance)  --> * I16: - P(f_1(c_1) c_1)  + =(c_1 f_3(c_1))
                          + =(f_1(c_1) c_1)
T8 (instance)  --> * I17: + P(c_2 c_2)  - =(c_2 f_4(c_2))  - =(c_2 c_3)
T8 (instance)  --> * I18: + P(c_2 c_1)  - =(c_2 f_4(c_1))  - =(c_1 c_3)
T8 (instance)  --> * I19: + P(c_1 c_2)  - =(c_1 f_4(c_2))  - =(c_2 c_3)
T8 (instance)  --> * I20: + P(c_1 c_1)  - =(c_1 f_4(c_1))  - =(c_1 c_3)
T9 (instance)  --> * I21: - P(c_2 c_2)  + =(c_2 f_4(c_2))  - =(c_2 c_3)
T9 (instance)  --> * I22: - P(c_2 c_1)  + =(c_2 f_4(c_1))  - =(c_1 c_3)
T9 (instance)  --> * I23: - P(c_1 c_2)  + =(c_1 f_4(c_2))  - =(c_2 c_3)
T9 (instance)  --> * I24: - P(c_1 c_1)  + =(c_1 f_4(c_1))  - =(c_1 c_3)
T10 (instance) --> * I25: - P(f_5(c_2 c_2) c_2)  - =(f_5(c_2 c_2) c_2)
                          + =(c_2 c_3)
T10 (instance) --> * I26: - P(f_5(c_2 c_1) c_1)  - =(f_5(c_2 c_1) c_2)
                          + =(c_1 c_3)
T10 (instance) --> * I27: - P(f_5(c_1 c_2) c_2)  - =(f_5(c_1 c_2) c_1)
                          + =(c_2 c_3)
T10 (instance) --> * I28: - P(f_5(c_1 c_1) c_1)  - =(f_5(c_1 c_1) c_1)
                          + =(c_1 c_3)
T11 (instance) --> * I29: + P(f_5(c_2 c_2) c_2)  + =(f_5(c_2 c_2) c_2)
                          + =(c_2 c_3)
T11 (instance) --> * I30: + P(f_5(c_2 c_1) c_1)  + =(f_5(c_2 c_1) c_2)
                          + =(c_1 c_3)
T11 (instance) --> * I31: + P(f_5(c_1 c_2) c_2)  + =(f_5(c_1 c_2) c_1)
                          + =(c_2 c_3)
T11 (instance) --> * I32: + P(f_5(c_1 c_1) c_1)  + =(f_5(c_1 c_1) c_1)
                          + =(c_1 c_3)

Splitpart 2      analogous to splitpart 1

Refutation of Splitpart 1:
==========================

I23,2 & I21,2     --> * P65:    + =(c_2 c_1)  - P(c_1 c_2)  - =(c_2 c_3)
                                - P(c_2 c_2)  - =(c_2 c_3)
P65 3=5           --> * D66:    + =(c_2 c_1)  - P(c_1 c_2)
                                - =(c_2 c_3)  - P(c_2 c_2)
D66,1 & A2,1      --> * R67:    - P(c_1 c_2)  - =(c_2 c_3)  - P(c_2 c_2)
I24,2 & I22,2     --> * P68:    + =(c_2 c_1)  - P(c_1 c_1)  - =(c_1 c_3)
                                - P(c_2 c_1)  - =(c_1 c_3)
P68 3=5           --> * D69:    + =(c_2 c_1)  - P(c_1 c_1)
                                - =(c_1 c_3)  - P(c_2 c_1)
D69,1 & A2,1      --> * R70:    - P(c_1 c_1)  - =(c_1 c_3)  - P(c_2 c_1)
A3,1 & I19,2      --> * P71:    - =(c_1 c_1)  + =(f_4(c_2) c_2)
                                + P(c_1 c_2)  - =(c_2 c_3)
P71,1 & A1,1      --> * R72:    + =(f_4(c_2) c_2)  + P(c_1 c_2)  - =(c_2 c_3)
R72,1 & I17,2     --> * P74:    - =(c_2 c_2)  + P(c_1 c_2)  - =(c_2 c_3)
                                + P(c_2 c_2)  - =(c_2 c_3)
P74 3=5           --> * D75:    - =(c_2 c_2)  + P(c_1 c_2)
                                - =(c_2 c_3)  + P(c_2 c_2)
D75,1 & A1,1      --> * R76:    + P(c_1 c_2)  - =(c_2 c_3)  + P(c_2 c_2)
A3,1 & I20,2      --> * P77:    - =(c_1 c_1)  + =(f_4(c_1) c_2)
                                + P(c_1 c_1)  - =(c_1 c_3)
P77,1 & A1,1      --> * R78:    + =(f_4(c_1) c_2)  + P(c_1 c_1)  - =(c_1 c_3)
R78,1 & I18,2     --> * P80:    - =(c_2 c_2)  + P(c_1 c_1)  - =(c_1 c_3)
                                + P(c_2 c_1)  - =(c_1 c_3)
```

```
P80 3=5          --> * D81:    - =(c_2 c_2)  + P(c_1 c_1)
                               - =(c_1 c_3)  + P(c_2 c_1)
D81,1 & A1,1     --> * R82:    + P(c_1 c_1)  - =(c_1 c_3)  + P(c_2 c_1)
A3,1 & I29,1     --> * P83:    + P(c_1 c_2)  + =(f_5(c_2 c_2) c_2)
                               + =(f_5(c_2 c_2) c_2)  + =(c_2 c_3)
P83 2=3          --> * D84:    + P(c_1 c_2)  + =(f_5(c_2 c_2) c_2)
                               + =(c_2 c_3)
A3,1 & I30,1     --> * P85:    + P(c_1 c_1)  + =(f_5(c_2 c_1) c_2)
                               + =(f_5(c_2 c_1) c_2)  + =(c_1 c_3)
P85 2=3          --> * D86:    + P(c_1 c_1)  + =(f_5(c_2 c_1) c_2)
                               + =(c_1 c_3)
A3,1 & I10,2     --> * P87:    - =(c_1 c_1)  + =(f_3(c_2) c_2)
                               + P(f_1(c_2) c_1)  + =(f_1(c_2) c_2)
P87,1 & A1,1     --> * R88:    + =(f_3(c_2) c_2)  + P(f_1(c_2) c_1)
                               + =(f_1(c_2) c_2)
A3,1 & I12,2     --> * P90:    - =(c_1 c_1)  + =(f_3(c_1) c_2)
                               + P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
P90,1 & A1,1     --> * R91:    + =(f_3(c_1) c_2)  + P(f_1(c_1) c_1)
                               + =(f_1(c_1) c_1)
I14,2 & I13,2    --> * P93:    + =(c_2 c_1)  - P(f_1(c_2) c_1)
                               + =(f_1(c_2) c_2)  - P(f_1(c_2) c_2)
                               + =(f_1(c_2) c_2)
P93 3=5          --> * D94:    + =(c_2 c_1)  - P(f_1(c_2) c_1)
                               + =(f_1(c_2) c_2)  - P(f_1(c_2) c_2)
D94,1 & A2,1     --> * R95:    - P(f_1(c_2) c_1)  + =(f_1(c_2) c_2)
                               - P(f_1(c_2) c_2)
A3,1 & R95,1     --> * P96:    - P(c_1 c_1)  + =(f_1(c_2) c_2)
                               + =(f_1(c_2) c_2)  - P(f_1(c_2) c_2)
P96 2=3          --> * D97:    - P(c_1 c_1)  + =(f_1(c_2) c_2)
                               - P(f_1(c_2) c_2)
A3,1 & D97,3     --> * P98:    - P(c_1 c_2)  + =(f_1(c_2) c_2)
                               - P(c_1 c_1)  + =(f_1(c_2) c_2)
P98 2=4          --> * D99:    - P(c_1 c_2)  + =(f_1(c_2) c_2)  - P(c_1 c_1)
I16,2 & I15,2    --> * P100:   + =(c_2 c_1)  - P(f_1(c_1) c_1)
                               + =(f_1(c_1) c_1)  - P(f_1(c_1) c_2)
                               + =(f_1(c_1) c_1)
P100 3=5         --> * D101:   + =(c_2 c_1)  - P(f_1(c_1) c_1)
                               + =(f_1(c_1) c_1)  - P(f_1(c_1) c_2)
D101,1 & A2,1    --> * R102:   - P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
                               - P(f_1(c_1) c_2)
R88,1 & I9,2     --> * P103:   - =(c_2 c_2)  + P(f_1(c_2) c_1)
                               + =(f_1(c_2) c_2)  + P(f_1(c_2) c_2)
                               + =(f_1(c_2) c_2)
P103 3=5         --> * D104:   - =(c_2 c_2)  + P(f_1(c_2) c_1)
                               + =(f_1(c_2) c_2)  + P(f_1(c_2) c_2)
D104,1 & A1,1    --> * R105:   + P(f_1(c_2) c_1)  + =(f_1(c_2) c_2)
                               + P(f_1(c_2) c_2)
A3,1 & R105,1    --> * P106:   + P(c_1 c_1)  + =(f_1(c_2) c_2)
                               + =(f_1(c_2) c_2)  + P(f_1(c_2) c_2)
P106 2=3         --> * D107:   + P(c_1 c_1)  + =(f_1(c_2) c_2)
                               + P(f_1(c_2) c_2)
A3,1 & D107,3    --> * P108:   + P(c_1 c_2)  + =(f_1(c_2) c_2)
                               + P(c_1 c_1)  + =(f_1(c_2) c_2)
P108 2=4         --> * D109:   + P(c_1 c_2)  + =(f_1(c_2) c_2)  + P(c_1 c_1)
R91,1 & I11,2    --> * P110:   - =(c_2 c_2)  + P(f_1(c_1) c_1)
                               + =(f_1(c_1) c_1)  + P(f_1(c_1) c_2)
                               + =(f_1(c_1) c_1)
P110 3=5         --> * D111:   - =(c_2 c_2)  + P(f_1(c_1) c_1)
                               + =(f_1(c_1) c_1)  + P(f_1(c_1) c_2)
D111,1 & A1,1    --> * R112:   + P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
                               + P(f_1(c_1) c_2)
A3,1 & I5,1      --> * P113:   + P(f_1(c_2) c_1)  + =(f_2(c_2 c_2) c_2)
                               + =(f_2(c_2 c_2) c_2)  - =(f_1(c_2) c_2)
P113 2=3         --> * D114:   + P(f_1(c_2) c_1)  + =(f_2(c_2 c_2) c_2)
                               - =(f_1(c_2) c_2)
```

```
A3,1 & I7,1       --> * P115:   + P(f_1(c_1) c_1)  + =(f_2(c_2 c_1) c_2)
                                + =(f_2(c_2 c_1) c_2)  - =(f_1(c_1) c_1)
P115 2=3          --> * D116:   + P(f_1(c_1) c_1)  + =(f_2(c_2 c_1) c_2)
                                - =(f_1(c_1) c_1)
D84,2 & I25,1     --> * P117:   - P(c_2 c_2)  + P(c_1 c_2)  + =(c_2 c_3)
                                - =(f_5(c_2 c_2) c_2)  + =(c_2 c_3)
P117 3=5          --> * D118:   - P(c_2 c_2)  + P(c_1 c_2)
                                + =(c_2 c_3)  - =(f_5(c_2 c_2) c_2)
D118,4 & D84,2    --> * R119:   - P(c_2 c_2)  + P(c_1 c_2)  + =(c_2 c_3)
                                + P(c_1 c_2)  + =(c_2 c_3)
R119 2=4          --> * D120:   - P(c_2 c_2)  + P(c_1 c_2)
                                + =(c_2 c_3)  + =(c_2 c_3)
D120 3=4          --> * D121:   - P(c_2 c_2)  + P(c_1 c_2)  + =(c_2 c_3)
D86,2 & I26,1     --> * P122:   - P(c_2 c_1)  + P(c_1 c_1)  + =(c_1 c_3)
                                - =(f_5(c_2 c_1) c_2)  + =(c_1 c_3)
P122 3=5          --> * D123:   - P(c_2 c_1)  + P(c_1 c_1)
                                + =(c_1 c_3)  - =(f_5(c_2 c_1) c_2)
D123,4 & D86,2    --> * R124:   - P(c_2 c_1)  + P(c_1 c_1)  + =(c_1 c_3)
                                + P(c_1 c_1)  + =(c_1 c_3)
R124 2=4          --> * D125:   - P(c_2 c_1)  + P(c_1 c_1)
                                + =(c_1 c_3)  + =(c_1 c_3)
D125 3=4          --> * D126:   - P(c_2 c_1)  + P(c_1 c_1)  + =(c_1 c_3)
A3,1 & I27,1      --> * P127:   - P(c_1 c_2)  + =(f_5(c_1 c_2) c_2)
                                - =(f_5(c_1 c_2) c_1)  + =(c_2 c_3)
P127,3 & A3,1     --> * R128:   - P(c_1 c_2)  + =(f_5(c_1 c_2) c_2)
                                + =(c_2 c_3)  + =(f_5(c_1 c_2) c_2)
R128 2=4          --> * D129:   - P(c_1 c_2)  + =(f_5(c_1 c_2) c_2)
                                + =(c_2 c_3)
D129,2 & I31,1    --> * P138:   + P(c_2 c_2)  - P(c_1 c_2)  + =(c_2 c_3)
                                + =(f_5(c_1 c_2) c_1)  + =(c_2 c_3)
P138 3=5          --> * D139:   + P(c_2 c_2)  - P(c_1 c_2)
                                + =(c_2 c_3)  + =(f_5(c_1 c_2) c_1)
D139,4 & D129     --> * RW140:  + P(c_2 c_2)  - P(c_1 c_2)
                                + =(c_2 c_3)  + =(c_2 c_1)
RW140,4 & A2,1    --> * R141:   + P(c_2 c_2)  - P(c_1 c_2)  + =(c_2 c_3)
A3,1 & I28,1      --> * P148:   - P(c_1 c_1)  + =(f_5(c_1 c_1) c_2)
                                - =(f_5(c_1 c_1) c_1)  + =(c_1 c_3)
P148,3 & A3,1     --> * R149:   - P(c_1 c_1)  + =(f_5(c_1 c_1) c_2)
                                + =(c_1 c_3)  + =(f_5(c_1 c_1) c_2)
R149 2=4          --> * D150:   - P(c_1 c_1)  + =(f_5(c_1 c_1) c_2)
                                + =(c_1 c_3)
D150,2 & I32,1    --> * P159:   + P(c_2 c_1)  - P(c_1 c_1)  + =(c_1 c_3)
                                + =(f_5(c_1 c_1) c_1)  + =(c_1 c_3)
P159 3=5          --> * D160:   + P(c_2 c_1)  - P(c_1 c_1)
                                + =(c_1 c_3)  + =(f_5(c_1 c_1) c_1)
D160,4 & D150     --> * RW161:  + P(c_2 c_1)  - P(c_1 c_1)
                                + =(c_1 c_3)  + =(c_2 c_1)
RW161,4 & A2,1    --> * R162:   + P(c_2 c_1)  - P(c_1 c_1)  + =(c_1 c_3)
D114,2 & I1,1     --> * P169:   - P(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  - =(f_2(c_2 c_2) c_2)
                                - =(f_1(c_2) c_2)
P169 3=5          --> * D170:   - P(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  - =(f_2(c_2 c_2) c_2)
D170,4 & D114,2   --> * R171:   - P(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)
R171 2=4          --> * D172:   - P(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  - =(f_1(c_2) c_2)
D172 3=4          --> * D173:   - P(f_1(c_2) c_2)  + P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)
D116,2 & I3,1     --> * P174:   - P(f_1(c_1) c_2)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  - =(f_2(c_2 c_1) c_2)
                                - =(f_1(c_1) c_1)
P174 3=5          --> * D175:   - P(f_1(c_1) c_2)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  - =(f_2(c_2 c_1) c_2)
```

```
D175,4 & D116,2   --> * R176:   - P(f_1(c_1) c_2)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)
R176 2=4          --> * D177:   - P(f_1(c_1) c_2)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  - =(f_1(c_1) c_1)
D177 3=4          --> * D178:   - P(f_1(c_1) c_2)  + P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)
A3,1 & D178,2     --> * P179:   + P(c_1 c_1)  + =(f_1(c_1) c_2)
                                - P(f_1(c_1) c_2)  - =(f_1(c_1) c_1)
P179,4 & A3,1     --> * R180:   + P(c_1 c_1)  + =(f_1(c_1) c_2)
                                - P(f_1(c_1) c_2)  + =(f_1(c_1) c_2)
R180 2=4          --> * D181:   + P(c_1 c_1)  + =(f_1(c_1) c_2)
                                - P(f_1(c_1) c_2)
A3,1 & D181,3     --> * P182:   - P(c_1 c_2)  + =(f_1(c_1) c_2)
                                + P(c_1 c_1)  + =(f_1(c_1) c_2)
P182 2=4          --> * D183:   - P(c_1 c_2)  + =(f_1(c_1) c_2)  + P(c_1 c_1)
A3,1 & I2,1       --> * P184:   - P(f_1(c_2) c_1)  + =(f_2(c_1 c_2) c_2)
                                - =(f_2(c_1 c_2) c_1)  - =(f_1(c_2) c_2)
P184,3 & A3,1     --> * R185:   - P(f_1(c_2) c_1)  + =(f_2(c_1 c_2) c_2)
                                - =(f_1(c_2) c_2)  + =(f_2(c_1 c_2) c_2)
R185 2=4          --> * D186:   - P(f_1(c_2) c_1)  + =(f_2(c_1 c_2) c_2)
                                - =(f_1(c_2) c_2)
D186,2 & I6,1     --> * P195:   + P(f_1(c_2) c_2)  - P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  + =(f_2(c_1 c_2) c_1)
                                - =(f_1(c_2) c_2)
P195 3=5          --> * D196:   + P(f_1(c_2) c_2)  - P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  + =(f_2(c_1 c_2) c_1)
D196,4 & D186     --> * RW197:  + P(f_1(c_2) c_2)  - P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)  + =(c_2 c_1)
RW197,4 & A2,1    --> * R198:   + P(f_1(c_2) c_2)  - P(f_1(c_2) c_1)
                                - =(f_1(c_2) c_2)
A3,1 & I4,1       --> * P226:   - P(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_2)
                                - =(f_2(c_1 c_1) c_1)  - =(f_1(c_1) c_1)
P226,3 & A3,1     --> * R227:   - P(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_2)
                                - =(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_2)
R227 2=4          --> * D228:   - P(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_2)
                                - =(f_1(c_1) c_1)
D228,2 & I8,1     --> * P237:   + P(f_1(c_1) c_2)  - P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_1)
                                - =(f_1(c_1) c_1)
P237 3=5          --> * D238:   + P(f_1(c_1) c_2)  - P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  + =(f_2(c_1 c_1) c_1)
D238,4 & D228     --> * RW239:  + P(f_1(c_1) c_2)  - P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)  + =(c_2 c_1)
RW239,4 & A2,1    --> * R240:   + P(f_1(c_1) c_2)  - P(f_1(c_1) c_1)
                                - =(f_1(c_1) c_1)
A3,1 & R240,2     --> * P241:   - P(c_1 c_1)  + =(f_1(c_1) c_2)
                                + P(f_1(c_1) c_2)  - =(f_1(c_1) c_1)
P241,4 & A3,1     --> * R242:   - P(c_1 c_1)  + =(f_1(c_1) c_2)
                                + P(f_1(c_1) c_2)  + =(f_1(c_1) c_2)
R242 2=4          --> * D243:   - P(c_1 c_1)  + =(f_1(c_1) c_2)
                                + P(f_1(c_1) c_2)
A3,1 & D243,3     --> * P244:   + P(c_1 c_2)  + =(f_1(c_1) c_2)
                                - P(c_1 c_1)  + =(f_1(c_1) c_2)
P244 2=4          --> * D245:   + P(c_1 c_2)  + =(f_1(c_1) c_2)  - P(c_1 c_1)
D109,2 & D173,2   --> * P260:   + P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                - P(f_1(c_2) c_2)  - =(f_1(c_2) c_2)
P260,5 & D109,2   --> * R261:   + P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                - P(f_1(c_2) c_2)  + P(c_1 c_2)  + P(c_1 c_1)
R261 2=5          --> * D262:   + P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                - P(f_1(c_2) c_2)  + P(c_1 c_1)
D262 3=5          --> * D263:   + P(c_2 c_1)  + P(c_1 c_2)
                                + P(c_1 c_1)  - P(f_1(c_2) c_2)
D263,4 & D109     --> * RW264:  + P(c_2 c_1)  + P(c_1 c_2)
                                + P(c_1 c_1)  - P(c_2 c_2)
```

```
RW264,3 & R162,2  --> * R265:   + P(c_2 c_1)  + P(c_1 c_2)  - P(c_2 c_2)
                                + P(c_2 c_1)  + =(c_1 c_3)
R265 4=1          --> * D266:   + P(c_1 c_2)  - P(c_2 c_2)
                                + P(c_2 c_1)  + =(c_1 c_3)
D266,1 & R67,1    --> * R267:   - P(c_2 c_2)  + P(c_2 c_1)  + =(c_1 c_3)
                                - =(c_2 c_3)  - P(c_2 c_2)
R267 5=1          --> * D268:   + P(c_2 c_1)  + =(c_1 c_3)
                                - =(c_2 c_3)  - P(c_2 c_2)
D268,3 & A3,2     --> * R269:   + P(c_2 c_1)  + =(c_1 c_3)
                                - P(c_2 c_2)  + =(c_3 c_1)
R269 2=4          --> * D270:   + P(c_2 c_1)  + =(c_1 c_3)  - P(c_2 c_2)
D99,2 & D173,2    --> * P271:   + P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                - P(f_1(c_2) c_2)  - =(f_1(c_2) c_2)
P271,5 & D97,2    --> * R272:   + P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                - P(f_1(c_2) c_2)  - P(c_1 c_1)
                                - P(f_1(c_2) c_2)
R272 3=5          --> * D273:   + P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                - P(f_1(c_2) c_2)  - P(f_1(c_2) c_2)
D273 4=5          --> * D274:   + P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_1 c_1)  - P(f_1(c_2) c_2)
D274,4 & D99      --> * RW275:  + P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_1 c_1)  - P(c_2 c_2)
R82,1 & RW275,3   --> * R276:   - =(c_1 c_3)  + P(c_2 c_1)  + P(c_2 c_1)
                                - P(c_1 c_2)  - P(c_2 c_2)
R276 3=2          --> * D277:   - =(c_1 c_3)  + P(c_2 c_1)
                                - P(c_1 c_2)  - P(c_2 c_2)
D277,1 & D270,2   --> * R278:   + P(c_2 c_1)  - P(c_1 c_2)  - P(c_2 c_2)
                                + P(c_2 c_1)  - P(c_2 c_2)
R278 1=4          --> * D279:   + P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_2 c_2)  - P(c_2 c_2)
D279 3=4          --> * D280:   + P(c_2 c_1)  - P(c_1 c_2)  - P(c_2 c_2)
RW264,2 & D280,2  --> * R281:   + P(c_2 c_1)  + P(c_1 c_1)  - P(c_2 c_2)
                                + P(c_2 c_1)  - P(c_2 c_2)
R281 1=4          --> * D282:   + P(c_2 c_1)  + P(c_1 c_1)
                                - P(c_2 c_2)  - P(c_2 c_2)
D282 3=4          --> * D283:   + P(c_2 c_1)  + P(c_1 c_1)  - P(c_2 c_2)
D183,2 & R102,1   --> * P296:   - P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(f_1(c_1) c_1)  - P(f_1(c_1) c_2)
P296,4 & D183     --> * RW297:  - P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(c_2 c_1)  - P(f_1(c_1) c_2)
RW297,5 & D183    --> * RW298:  - P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(c_2 c_1)  - P(c_2 c_2)
RW298,4 & A2,1    --> * R299:   - P(c_2 c_1)  - P(c_1 c_2)
                                + P(c_1 c_1)  - P(c_2 c_2)
R299,1 & D283,1   --> * R300:   - P(c_1 c_2)  + P(c_1 c_1)  - P(c_2 c_2)
                                + P(c_1 c_1)  - P(c_2 c_2)
R300 2=4          --> * D301:   - P(c_1 c_2)  + P(c_1 c_1)
                                - P(c_2 c_2)  - P(c_2 c_2)
D301 3=4          --> * D302:   - P(c_1 c_2)  + P(c_1 c_1)  - P(c_2 c_2)
D302,2 & R70,1    --> * R303:   - P(c_1 c_2)  - P(c_2 c_2)
                                - =(c_1 c_3)  - P(c_2 c_1)
R303,4 & D280,1   --> * R304:   - P(c_1 c_2)  - P(c_2 c_2)  - =(c_1 c_3)
                                - P(c_1 c_2)  - P(c_2 c_2)
R304 1=4          --> * D305:   - P(c_1 c_2)  - P(c_2 c_2)
                                - =(c_1 c_3)  - P(c_2 c_2)
D305 2=4          --> * D306:   - P(c_1 c_2)  - P(c_2 c_2)  - =(c_1 c_3)
D121,2 & D306,1   --> * R307:   - P(c_2 c_2)  + =(c_2 c_3)
                                - P(c_2 c_2)  - =(c_1 c_3)
R307 3=1          --> * D308:   + =(c_2 c_3)  - P(c_2 c_2)  - =(c_1 c_3)
D308,3 & A3,1     --> * R309:   + =(c_2 c_3)  - P(c_2 c_2)  + =(c_3 c_2)
R309 1=3          --> * D310:   + =(c_2 c_3)  - P(c_2 c_2)
R141,1 & D310,2   --> * R311:   - P(c_1 c_2)  + =(c_2 c_3)  + =(c_2 c_3)
R311 2=3          --> * D312:   - P(c_1 c_2)  + =(c_2 c_3)
R67,2 & D310,1    --> * R315:   - P(c_1 c_2)  - P(c_2 c_2)  - P(c_2 c_2)
R315 2=3          --> * D316:   - P(c_1 c_2)  - P(c_2 c_2)
```

```
D183,2 & R112,1   --> * P323:   + P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(f_1(c_1) c_1)  + P(f_1(c_1) c_2)
P323,4 & D183     --> * RW324:  + P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(c_2 c_1)  + P(f_1(c_1) c_2)
RW324,5 & D183    --> * RW325:  + P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(c_2 c_1)  + P(c_2 c_2)
RW325,5 & D316,2  --> * R326:   + P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
                                + =(c_2 c_1)  - P(c_1 c_2)
R326 2=5          --> * D327:   + P(c_2 c_1)  - P(c_1 c_2)
                                + P(c_1 c_1)  + =(c_2 c_1)
D327,4 & A2,1     --> * R328:   + P(c_2 c_1)  - P(c_1 c_2)  + P(c_1 c_1)
R328,3 & R162,2   --> * R329:   + P(c_2 c_1)  - P(c_1 c_2)
                                + P(c_2 c_1)  + =(c_1 c_3)
R329 3=1          --> * D330:   - P(c_1 c_2)  + P(c_2 c_1)  + =(c_1 c_3)
R76,1 & D330,1    --> * R331:   - =(c_2 c_3)  + P(c_2 c_2)
                                + P(c_2 c_1)  + =(c_1 c_3)
R331,2 & D270,3   --> * R332:   - =(c_2 c_3)  + P(c_2 c_1)  + =(c_1 c_3)
                                + P(c_2 c_1)  + =(c_1 c_3)
R332 2=4          --> * D333:   - =(c_2 c_3)  + P(c_2 c_1)
                                + =(c_1 c_3)  + =(c_1 c_3)
D333 3=4          --> * D334:   - =(c_2 c_3)  + P(c_2 c_1)  + =(c_1 c_3)
D334,1 & A3,2     --> * R335:   + P(c_2 c_1)  + =(c_1 c_3)  + =(c_3 c_1)
R335 2=3          --> * D336:   + P(c_2 c_1)  + =(c_1 c_3)
R82,2 & D336,2    --> * R339:   + P(c_1 c_1)  + P(c_2 c_1)  + P(c_2 c_1)
R339 2=3          --> * D340:   + P(c_1 c_1)  + P(c_2 c_1)
D126,1 & D336,1   --> * R341:   + P(c_1 c_1)  + =(c_1 c_3)  + =(c_1 c_3)
R341 2=3          --> * D342:   + P(c_1 c_1)  + =(c_1 c_3)
D109,2 & R198,2   --> * P354:   - P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                + P(f_1(c_2) c_2)  - =(f_1(c_2) c_2)
P354,5 & D107,2   --> * R355:   - P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                + P(f_1(c_2) c_2)  + P(c_1 c_1)
                                + P(f_1(c_2) c_2)
R355 3=5          --> * D356:   - P(c_2 c_1)  + P(c_1 c_2)  + P(c_1 c_1)
                                + P(f_1(c_2) c_2)  + P(f_1(c_2) c_2)
D356 4=5          --> * D357:   - P(c_2 c_1)  + P(c_1 c_2)
                                + P(c_1 c_1)  + P(f_1(c_2) c_2)
D357,1 & D340,2   --> * R358:   + P(c_1 c_2)  + P(c_1 c_1)
                                + P(f_1(c_2) c_2)  + P(c_1 c_1)
R358 2=4          --> * D359:   + P(c_1 c_2)  + P(c_1 c_1)  + P(f_1(c_2) c_2)
D359,3 & D109     --> * RW360:  + P(c_1 c_2)  + P(c_1 c_1)  + P(c_2 c_2)
RW360,2 & R70,1   --> * R361:   + P(c_1 c_2)  + P(c_2 c_2)
                                - =(c_1 c_3)  - P(c_2 c_1)
D99,2 & R198,2    --> * P367:   - P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                + P(f_1(c_2) c_2)  - =(f_1(c_2) c_2)
P367,5 & D99,2    --> * R368:   - P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                + P(f_1(c_2) c_2)  - P(c_1 c_2)  - P(c_1 c_1)
R368 2=5          --> * D369:   - P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
                                + P(f_1(c_2) c_2)  - P(c_1 c_1)
D369 3=5          --> * D370:   - P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_1 c_1)  + P(f_1(c_2) c_2)
D370,4 & D99      --> * RW371:  - P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_1 c_1)  + P(c_2 c_2)
RW371,4 & D316,2  --> * R372:   - P(c_2 c_1)  - P(c_1 c_2)
                                - P(c_1 c_1)  - P(c_1 c_2)
R372 2=4          --> * D373:   - P(c_2 c_1)  - P(c_1 c_2)  - P(c_1 c_1)
D342,1 & D373,3   --> * R374:   + =(c_1 c_3)  - P(c_2 c_1)  - P(c_1 c_2)
R374,2 & D336,1   --> * R375:   + =(c_1 c_3)  - P(c_1 c_2)  + =(c_1 c_3)
R375 1=3          --> * D376:   + =(c_1 c_3)  - P(c_1 c_2)
R76,1 & D376,2    --> * R377:   - =(c_2 c_3)  + P(c_2 c_2)  + =(c_1 c_3)
R377,1 & A3,2     --> * R378:   + P(c_2 c_2)  + =(c_1 c_3)  + =(c_3 c_1)
R378 2=3          --> * D379:   + P(c_2 c_2)  + =(c_1 c_3)
R361,3 & D379,2   --> * R380:   + P(c_1 c_2)  + P(c_2 c_2)
                                - P(c_2 c_1)  + P(c_2 c_2)
R380 2=4          --> * D381:   + P(c_1 c_2)  + P(c_2 c_2)  - P(c_2 c_1)
D381,1 & D379     --> * RW382:  + P(c_3 c_2)  + P(c_2 c_2)  - P(c_2 c_1)
```

```
RW382,3 & D379    --> * RW383:  + P(c_3 c_2)  + P(c_2 c_2)  - P(c_2 c_3)
RW360,1 & D379    --> * RW384:  + P(c_3 c_2)  + P(c_1 c_1)  + P(c_2 c_2)
RW384,2 & D379    --> * RW385:  + P(c_3 c_2)  + P(c_3 c_1)  + P(c_2 c_2)
RW385,2 & D379    --> * RW386:  + P(c_3 c_2)  + P(c_3 c_3)  + P(c_2 c_2)
D379,2 & A2,1     --> * P388:   - =(c_2 c_3)  + P(c_2 c_2)
D379,2 & D312,1   --> * P395:   - P(c_3 c_2)  + P(c_2 c_2)  + =(c_2 c_3)
P395,3 & P388,1   --> * R396:   - P(c_3 c_2)  + P(c_2 c_2)  + P(c_2 c_2)
R396 2=3          --> * D397:   - P(c_3 c_2)  + P(c_2 c_2)
RW386,1 & D397,1  --> * R398:   + P(c_3 c_3)  + P(c_2 c_2)  + P(c_2 c_2)
R398 2=3          --> * D399:   + P(c_3 c_3)  + P(c_2 c_2)
RW383,1 & D397,1  --> * R400:   + P(c_2 c_2)  - P(c_2 c_3)  + P(c_2 c_2)
R400 1=3          --> * D401:   + P(c_2 c_2)  - P(c_2 c_3)
D399,2 & D310,2   --> * R402:   + P(c_3 c_3)  + =(c_2 c_3)
D399,2 & R402     --> * RW403:  + P(c_3 c_3)  + P(c_3 c_2)
RW403,2 & R402    --> * RW404:  + P(c_3 c_3)  + P(c_3 c_3)
RW404 (instance)  --> * I405:   + P(c_3 c_3)  + P(c_3 c_3)
I405 1=2          --> * D406:   + P(c_3 c_3)
D379,2 & D245,2   --> * P449:   + =(f_1(c_3) c_2)  + P(c_2 c_2)
                                + P(c_1 c_2)  - P(c_1 c_1)
P449,3 & D379     --> * RW450:  + =(f_1(c_3) c_2)  + P(c_2 c_2)
                                + P(c_3 c_2)  - P(c_1 c_1)
RW450,4 & D379    --> * RW451:  + =(f_1(c_3) c_2)  + P(c_2 c_2)
                                + P(c_3 c_2)  - P(c_3 c_1)
RW451,4 & D379    --> * RW452:  + =(f_1(c_3) c_2)  + P(c_2 c_2)
                                + P(c_3 c_2)  - P(c_3 c_3)
RW452,4 & D406,1  --> * R453:   + =(f_1(c_3) c_2)  + P(c_2 c_2)  + P(c_3 c_2)
R453,3 & D397,1   --> * R454:   + =(f_1(c_3) c_2)  + P(c_2 c_2)  + P(c_2 c_2)
R454 2=3          --> * D455:   + =(f_1(c_3) c_2)  + P(c_2 c_2)
D379,2 & I11,2    --> * P504:   - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(f_1(c_1) c_2)  + =(f_1(c_1) c_1)
P504,3 & D379     --> * RW505:  - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(f_1(c_3) c_2)  + =(f_1(c_1) c_1)
RW505,3 & D455    --> * RW506:  - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(c_2 c_2)  + =(f_1(c_1) c_1)
RW506,4 & D379    --> * RW507:  - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(c_2 c_2)  + =(f_1(c_3) c_1)
RW507,4 & D379    --> * RW508:  - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(c_2 c_2)  + =(f_1(c_3) c_3)
RW508,4 & D455    --> * RW509:  - =(c_2 f_3(c_3))  + P(c_2 c_2)
                                + P(c_2 c_2)  + =(c_2 c_3)
RW509 3=2         --> * D510:   - =(c_2 f_3(c_3))  + P(c_2 c_2)  + =(c_2 c_3)
D510,3 & P388,1   --> * R511:   - =(c_2 f_3(c_3))  + P(c_2 c_2)  + P(c_2 c_2)
R511 2=3          --> * D512:   - =(c_2 f_3(c_3))  + P(c_2 c_2)
D379,2 & R91,1    --> * P538:   + =(f_3(c_3) c_2)  + P(c_2 c_2)
                                + P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
P538,1 & D512,1   --> * R539:   + P(c_2 c_2)  + P(f_1(c_1) c_1)
                                + =(f_1(c_1) c_1)  + P(c_2 c_2)
R539 1=4          --> * D540:   + P(c_2 c_2)  + P(f_1(c_1) c_1)
                                + =(f_1(c_1) c_1)
D540,2 & D379     --> * RW541:  + P(c_2 c_2)  + P(f_1(c_3) c_1)
                                + =(f_1(c_1) c_1)
RW541,2 & D379    --> * RW542:  + P(c_2 c_2)  + P(f_1(c_3) c_3)
                                + =(f_1(c_1) c_1)
RW542,2 & D455    --> * RW543:  + P(c_2 c_2)  + P(c_2 c_3)  + =(f_1(c_1) c_1)
RW543,3 & D379    --> * RW544:  + P(c_2 c_2)  + P(c_2 c_3)  + =(f_1(c_3) c_1)
RW544,3 & D455    --> * RW545:  + P(c_2 c_2)  + P(c_2 c_3)  + =(c_2 c_1)
RW545,3 & D379    --> * RW546:  + P(c_2 c_2)  + P(c_2 c_3)  + =(c_2 c_3)
RW546,3 & P388,1  --> * R547:   + P(c_2 c_2)  + P(c_2 c_3)  + P(c_2 c_2)
R547 1=3          --> * D548:   + P(c_2 c_2)  + P(c_2 c_3)
D548,2 & D401,2   --> * R549:   + P(c_2 c_2)  + P(c_2 c_2)
R549 1=2          --> * D550:   + P(c_2 c_2)
D316,2 & D550,1   --> * R552:   - P(c_1 c_2)
D310,2 & D550,1   --> * R553:   + =(c_2 c_3)
D245,1 & R552,1   --> * R554:   + =(f_1(c_1) c_2)  - P(c_1 c_1)
A2,1 & R553       --> * RW608:  - =(c_3 c_1)
```

```
R102,3 & R553     --> * RW663:  - P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
                                - P(f_1(c_1) c_3)
R554,1 & R553     --> * RW689:  + =(f_1(c_1) c_3)  - P(c_1 c_1)
D336,1 & R553     --> * RW692:  + P(c_3 c_1)  + =(c_1 c_3)
RW689,2 & D342,1  --> * R697:   + =(f_1(c_1) c_3)  + =(c_1 c_3)
R697,2 & RW608,1  --> * R698:   + =(f_1(c_1) c_3)
RW663,3 & R698    --> * RW719:  - P(f_1(c_1) c_1)  + =(f_1(c_1) c_1)
                                - P(c_3 c_3)
RW719,2 & R698    --> * RW720:  - P(f_1(c_1) c_1)  + =(c_3 c_1)  - P(c_3 c_3)
RW720,1 & R698    --> * RW721:  - P(c_3 c_1)  + =(c_3 c_1)  - P(c_3 c_3)
RW721,3 & D406,1  --> * R739:   - P(c_3 c_1)  + =(c_3 c_1)
R739,2 & RW608,1  --> * R742:   - P(c_3 c_1)
RW692,1 & R742,1  --> * R744:   + =(c_1 c_3)
R744,1 & RW608,1  --> * R745:   []

Refutation of Splitpart 2:
==========================

similar to splitpart 1
```

B.2.6 Example 54

Example 54 is a nice mathematical problem proved in the fifties [Mon55, She53]. The biggest problem with it was a typing error in F. Pelletier's publication. He wrote `IN(X K) and IN(Z Y)` in the last line of the axiom, which causes the unrefutability of the set of clauses.

```
Formulae Given to the Editor
============================

Axioms:   * Montague's Paradox of Grounded Classes *
          ALL Y (EX Z (ALL X IN(X Z) EQV X = Y))
          * Z = {Y} *
          ALL X REG(X)
                EQV (ALL K IN(X K)
                           IMPL (EX Y IN(Y K) AND
                                 NOT (EX Z IN(Z K) AND IN(Z Y))))

Theorems: * REG of Montague Doesn't Exist *
          NOT (EX W (ALL X IN(X W) EQV REG(X)))

Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x,y:Any + IN(y f_1(x))  - =(y x)
        * A3:   All x,y:Any - IN(y f_1(x))  + =(y x)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T4: All x:Any + IN(x c_1)  + IN(x f_3(x))
        * T5: All x,y:Any + IN(y c_1)  - IN(x f_3(y))
                          + IN(f_2(x y) f_3(y))
        * T6: All x,y:Any + IN(y c_1)  - IN(x f_3(y))  + IN(f_2(x y) x)
        * T7: All x,y:Any - IN(y c_1)  - IN(y x)  + IN(f_4(x y) x)
        * T8: All x,y,z:Any - IN(z c_1)  - IN(z y)  - IN(x y)
                            - IN(x f_4(y z))

Refutation:
===========

A1,1 & A2,2    --> * R1:   All x:Any + IN(x f_1(x))
```

```
T7,3 & A3,1    --> * R2:   All x,y:Any - IN(y c_1)  - IN(y f_1(x))
                                      + =(f_4(f_1(x) y) x)
R1,1 & R2,2    -->   R3:   All x:Any - IN(x c_1)  + =(f_4(f_1(x) x) x)
A2,1 & R2,2    --> * R4:   All x,y:Any - =(y x)  - IN(y c_1)
                                      + =(f_4(f_1(x) y) x)
R3,2 & R4      -->   RW5:  All x:Any - IN(x c_1)  + =(x x)
R1,1 & T8,2    --> * R6:   All x,y:Any - IN(y c_1)  - IN(x f_1(y))
                                      - IN(x f_4(f_1(y) y))
R6,3 & R4      --> * RW7:  All x,y:Any - IN(x c_1)  - IN(y f_1(x))  - IN(y x)
RW7 (factor)   --> * F8:   - IN(c_1 c_1)  - IN(c_1 f_1(c_1))
F8,2 & R1,1    --> * R9:   - IN(c_1 c_1)
T4,1 & R9,1    --> * R10:  + IN(c_1 f_2(c_1))
R10,1 & T6,2   --> * R11:  + IN(c_1 c_1)  + IN(f_3(c_1 c_1) c_1)
R11,1 & R9,1   --> * R12:  + IN(f_3(c_1 c_1) c_1)
T5,1 & R9,1    --> * R13:  - IN(c_1 f_2(c_1))  + IN(f_3(c_1 c_1) f_2(c_1))
R13,1 & R10,1  --> * R14:  + IN(f_3(c_1 c_1) f_2(c_1))
R14,1 & T7,2   --> * R15:  - IN(f_3(c_1 c_1) c_1)
                           + IN(f_4(f_2(c_1) f_3(c_1 c_1)) f_2(c_1))
R15,1 & R12,1  --> * R16:  + IN(f_4(f_2(c_1) f_3(c_1 c_1)) f_2(c_1))
R16,1 & T6,2   --> * R17:  + IN(c_1 c_1)
                           + IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1)
                                f_4(f_2(c_1) f_3(c_1 c_1)))
R17,1 & R9,1   --> * R18:  + IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1)
                                f_4(f_2(c_1) f_3(c_1 c_1)))
R16,1 & T5,2   --> * R19:  + IN(c_1 c_1)
                           + IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1) f_2(c_1))
R19,1 & R9,1   --> * R20:  + IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1) f_2(c_1))
R20,1 & T8,3   --> * R21:  - IN(f_3(c_1 c_1) c_1)  - IN(f_3(c_1 c_1) f_2(c_1))
                           - IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1)
                                f_4(f_2(c_1) f_3(c_1 c_1)))
R21,2 & R14,1  --> * R22:  - IN(f_3(c_1 c_1) c_1)
                           - IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1)
                                f_4(f_2(c_1) f_3(c_1 c_1)))
R22,1 & R12,1  --> * R23:  - IN(f_3(f_4(f_2(c_1) f_3(c_1 c_1)) c_1)
                                f_4(f_2(c_1) f_3(c_1 c_1)))
R23,1 & R18,1  --> * R24:  []
```

B.2.7 Example 55

Example 55 is a logical riddle given by L. Schubert. The question is: Who killed aunt Agatha? In refutational theorem proving we have to prove the answer: Aunt Agatha killed herself (`K(A A)`).

```
Formulae given to the editor
============================

Axioms:   EX X L(X) AND K(X A)
          L (A) AND L(B) AND L(C)
          ALL X L (X) IMPL X = A OR X = B OR X = C
          ALL Y,X K(X Y) IMPL H(X Y)
          ALL X,Y K(X Y) IMPL NOT R(X Y)
          ALL X H(A X) IMPL NOT H(C X)
          ALL X NOT X = B IMPL H(A X)
          ALL X NOT R(X A) IMPL H(B X)
          ALL X H(A X) IMPL H(B X)
          ALL X (EX Y NOT H(X Y))
          NOT A = B

Theorems: K(A A)

Set of Axiom Clauses Resulting from Normalization
=================================================

            A1:   All x:Any + =(x x)
          * A2:   + L(c_1)
```

```
        * A3:   + K(c_1 a)
          A4:   + L(a)
          A5:   + L(b)
          A6:   + L(c)
        * A7:   All x:Any - H(x f_1(x))
        * A8:   - =(a b)
        * A9:   All x,y:Any - K(y x)  + H(y x)
        * A10:  All x,y:Any - K(y x)  - R(y x)
        * A11:  All x:Any - H(a x)  - H(c x)
        * A12:  All x:Any + =(x b)  + H(a x)
        * A13:  All x:Any + R(x a)  + H(b x)
        * A14:  All x:Any - H(a x)  + H(b x)
        * A15:  All x:Any - L(x)  + =(x a)  + =(x b)  + =(x c)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T16: - K(a a)

Refutation:
===========

A2,1 & A15,1   --> * R5:    + =(c_1 a)  + =(c_1 b)  + =(c_1 c)
R5,1 & T16,1   --> * P6:    - K(c_1 a)  + =(c_1 b)  + =(c_1 c)
P6,1 & A3,1    --> * R7:    + =(c_1 b)  + =(c_1 c)
A12,2 & A11,1  --> * R12:   All x:Any + =(x b)  - H(c x)
A13,1 & A10,2  --> * R13:   All x:Any + H(b x)  - K(x a)
R13,1 & A7,1   --> * R14:   - K(f_1(b) a)
A12,2 & A14,1  --> * R17:   All x:Any + =(x b)  + H(b x)
R17,2 & A7,1   --> * R18:   + =(f_1(b) b)
R14,1 & R18    --> * RW19:  - K(b a)
R7,1 & RW19,1  --> * P20:   - K(c_1 a)  + =(c_1 c)
P20,1 & A3,1   --> * R21:   + =(c_1 c)
R12,2 & R21    --> * RW26:  All x:Any + =(x b)  - H(c_1 x)
RW26,2 & A9,2  --> * R27:   + =(a b)  - K(c_1 a)
R27,1 & A8,1   --> * R28:   - K(c_1 a)
R28,1 & A3,1   --> * R29:   []
```

B.2.8 Example 56

Example 56 is the first equational one of F. Pelletier using a function.

```
Formulae given to the editor
============================

Theorems: (ALL X (EX Y P(Y) AND X = F(Y)) IMPL P(X))
          EQV (ALL X P(X) IMPL P(F(X)))

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T1: All x:Any + =(x x)
        * T2: + P(c_1)  + P(c_3)
        * T3: + P(c_1)  - P(f(c_3))
        * T4: + =(c_2 f(c_1))  + P(c_3)
        * T5: + =(c_2 f(c_1))  - P(f(c_3))
        * T6: - P(c_2)  + P(c_3)
        * T7: - P(c_2)  - P(f(c_3))
        * T8: All x,y,z:Any - P(z)  - =(y f(z))  + P(y)
                            - P(x)  + P(f(x))

Initial Operations on Theorems
==============================

T8,2 & T1,1    --> * R1: All x:Any - P(x)  + P(f(x))  - P(x)  + P(f(x))
```

```
R1 3=1          --> * D2: All x:Any + P(f(x))  - P(x)  + P(f(x))
D2 3=1          --> * D3: All x:Any - P(x)  + P(f(x))

Refutation:
===========

D3,2 & T7,2   --> * R4:   - P(c_3)  - P(c_1)
R4,1 & T6,2   --> * R5:   - P(c_1)  - P(c_1)
R5 1=2        --> * D6:   - P(c_1)
D3,2 & T3,2   --> * R7:   - P(c_3)  + P(c_2)
R7,1 & T2,2   --> * R8:   + P(c_2)  + P(c_2)
R8 1=2        --> * D9:   + P(c_2)
D3,2 & T5,2   --> * R10:  - P(c_3)  + =(c_1 f(c_2))
R10,1 & T4,2  --> * R11:  + =(c_1 f(c_2))  + =(c_1 f(c_2))
R11 1=2       --> * D12:  + =(c_1 f(c_2))
D12,1 & D3,2  --> * P13:  + P(c_1)  - P(c_2)
P13,2 & D9,1  --> * R14:  + P(c_1)
R14,1 & D6,1  --> * R15:  []
```

B.2.9 Example 58

Example 58 checks pure equation application.

```
Formulae Given to the Editor
============================

Axioms:   ALL X,Y F(X) = G(Y)

Theorems: ALL X,Y F(F(X)) = F(G(Y))

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y:Any + =(f(y) g(x))

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T3: - =(f(f(c_2)) f(g(c_1)))

Refutation:
===========

A2,1 & T3,1  -->  * P1:  All x:Any - =(g(x) f(g(c_1)))
P1,1 & A2,1  -->  * R2:  []
```

B.2.10 Example 61

Example 61 checks whether the prover is able to deal with associativity.

```
Formulae given to the editor
============================

Axioms:   ALL X,Y,Z F(X F(Y Z)) = F(F(X Y) Z)

Theorems: ALL X,Y,Z,W F(X F(Y F(Z W))) = F(F(F(X Y) Z) W)

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(f(z f(y x)) f(f(z y) x))
```

```
Set of Theorem Clauses Resulting from Normalization
===================================================

        * T3: - =(f(c_4 f(c_2 f(c_3 c_1))) f(f(f(c_4 c_2) c_3) c_1))

Refutation:
===========

A2,1 & T3,1  -->  * P1:  - =(f(c_4 f(c_2 f(c_3 c_1)))
                            f(f(c_4 c_2) f(c_3 c_1)))
P1,1 & A2,1  -->  * R2:  []
```

B.2.11 Example 63

Example 63 is a simple group example.

```
Formulae Given to the Editor
============================

Axioms:   ALL X,Y,Z F(X F(Y Z)) = F(F(X Y) Z)
          ALL X F(A X) = X
          ALL X (EX Y F(Y X) = A)

Theorems: ALL X,Y,Z F(X Y) = F(Z Y) IMPL X = Z

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(f(z f(y x)) f(f(z y) x))
        * A3:   All x:Any + =(f(a x) x)
        * A4:   All x:Any + =(f(f_1(x) x) a)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T5: + =(f(c_3 c_1) f(c_2 c_1))
        * T6: - =(c_3 c_2)

Refutation:
===========

A4,1 & A2,1     -->  * P1:    All x,y:Any + =(f(f_1(y) f(y x)) f(a x))
P1,1 & A3       -->  * RW2:   All x,y:Any + =(f(f_1(y) f(y x)) x)
A4,1 & RW2,1    -->  * P4:    All x:Any + =(f(f_1(f_1(x)) a) x)
T5,1 & RW2,1    -->  * P10:   + =(f(f_1(c_3) f(c_2 c_1)) c_1)
P4,1 & RW2,1    -->  * P11:   All x:Any + =(f(f_1(f_1(f_1(x))) x) a)
P11,1 & RW2,1   -->  * P12:   All x:Any + =(f(f_1(f_1(f_1(f_1(x)))) a) x)
P12,1 & P4      -->  * RW13:  All x:Any + =(f_1(f_1(x)) x)
P4,1 & RW13     -->  * RW15:  All x:Any + =(f(x a) x)
RW13,1 & A4,1   -->  * P19:   All x:Any + =(f(x f_1(x)) a)
RW13,1 & RW2,1  -->  * P20:   All x,y:Any + =(f(y f(f_1(y) x)) x)
P19,1 & A2,1    -->  * P21:   All x,y:Any + =(f(y f(x f_1(f(y x)))) a)
P10,1 & P21,1   -->  * P22:   + =(f(f_1(c_3) f(f(c_2 c_1) f_1(c_1))) a)
P22,1 & A2      -->  * RW23:  + =(f(f_1(c_3) f(c_2 f(c_1 f_1(c_1)))) a)
RW23,1 & P19    -->  * RW24:  + =(f(f_1(c_3) f(c_2 a)) a)
RW24,1 & RW15   -->  * RW25:  + =(f(f_1(c_3) c_2) a)
RW25            -->  * RS26:  + =(a f(f_1(c_3) c_2))
RW15,1 & RS26   -->  * RW29:  All x:Any + =(f(x f(f_1(c_3) c_2)) x)
P20,1 & RW29,1  -->  * P35:   + =(c_2 c_3)
P35,1 & T6,1    -->  * R36:   []
```

B.2.12 Example 64

Example 64 is another simple group theorem.

```
Formulae given to the editor
============================

Axioms:   ALL X,Y,Z F(X F(Y Z)) = F(F(X Y) Z)
          ALL X F(A X) = X
          ALL X (EX Y F(Y X) = A)

Theorems: ALL X,Y F(Y X) = A IMPL F(X Y) = A

Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(f(z f(y x)) f(f(z y) x))
        * A3:   All x:Any + =(f(a x) x)
        * A4:   All x:Any + =(f(f_1(x) x) a)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T5: + =(f(c_1 c_2) a)
        * T6: - =(f(c_2 c_1) a)

Refutation:
===========

A4,1 & A2,1    -->  * P3:    All x,y:Any + =(f(f_1(y) f(y x)) f(a x))
P3,1 & A3      -->  * RW4:   All x,y:Any + =(f(f_1(y) f(y x)) x)
T5,1 & RW4,1   -->  * P5:    + =(f(f_1(c_1) a) c_2)
P5             -->  * RS6:   + =(c_2 f(f_1(c_1) a))
T6,1 & RS6     -->  * RW8:   - =(f(f(f_1(c_1) a) c_1) a)
RW8,1 & A2     -->  * RW13:  - =(f(f_1(c_1) f(a c_1)) a)
RW13,1 & A3    -->  * RW14:  - =(f(f_1(c_1) c_1) a)
RW14,1 & A4    -->  * RW15:  - =(a a)
RW15,1 & A1,1  -->  * R16:   []
```

B.2.13 Example 65

We omit example 65 because it is exactly the same as B.1.1.

B.2.14 Example 73, Four Pigeons – Three Holes

The general pigeon hole problem is: it is impossible that there are n holes and $n + 1$ pigeons, every pigeon is in a hole, and no hole contains two pigeons [CR79]. It is no problem to prove the general problem with arbitrary n [KP92], but of course the idea is to use the problem as a scheme to generate arbitrary complicated test examples.

The original formulation was in propositional logic. F. Pelletier proposes a series of corresponding problems specified using the equality predicate. The problem selected here is the instance for $n = 3$.

```
Formulae given to the editor
============================


Axioms:   EX X,Y,Z,W P(X) AND P(Y) AND P(Z) AND P(W)
                      AND NOT X = Y AND NOT X = Z AND NOT X = W
                      AND NOT Y = Z AND NOT Y = W AND NOT Z = W
          EX X,Y,Z H(X) AND H(Y) AND H(Z)
```

```
                      AND NOT X = Y AND NOT X = Z AND NOT Y = Z
                      AND ALL W H(W) IMPL (W = X OR W = Y OR W = Z)
          ALL X P(X) IMPL EX Y H(Y) AND In(X Y)
          ALL X
            H(X) IMPL ALL Y,Z P(Y) AND P(Z) AND In(Y X) AND In(Z X) IMPL Y = Z

Theorems: None

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   + P(c_1)
        * A3:   + P(c_2)
        * A4:   + P(c_3)
        * A5:   + P(c_4)
        * A6:   - =(c_1 c_2)
        * A7:   - =(c_1 c_3)
        * A8:   - =(c_1 c_4)
        * A9:   - =(c_2 c_3)
        * A10:  - =(c_2 c_4)
        * A11:  - =(c_3 c_4)
          A12:  + H(c_5)
          A13:  + H(c_6)
        * A14:  + H(c_7)
          A15:  - =(c_5 c_6)
          A16:  - =(c_5 c_7)
          A17:  - =(c_6 c_7)
        * A18:  All x:Any - P(x)  + H(f_1(x))
        * A19:  All x:Any - P(x)  + IN(x f_1(x))
        * A20:  All x:Any - H(x)  + =(x c_5)  + =(x c_6)  + =(x c_7)
        * A21:  All x,y,z:Any - H(z)  - P(y)  - P(x)  - IN(y z)  - IN(x z)
                             + =(y x)

Refutation:
===========

A18,2 & A20,1    --> * R1:    All x:Any - P(x)  + =(f_1(x) c_5)
                                        + =(f_1(x) c_6)  + =(f_1(x) c_7)
A5,1 & R1,1      --> * R2:    + =(f_1(c_4) c_5)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)
A4,1 & R1,1      --> * R3:    + =(f_1(c_3) c_5)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)
A3,1 & R1,1      --> * R4:    + =(f_1(c_2) c_5)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)
A2,1 & R1,1      --> * R5:    + =(f_1(c_1) c_5)  + =(f_1(c_1) c_6)
                              + =(f_1(c_1) c_7)
R2,1 & A19,2     --> * P6:    + IN(c_4 c_5)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  - P(c_4)
P6,4 & A5,1      --> * R7:    + IN(c_4 c_5)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)
R3,1 & A19,2     --> * P8:    + IN(c_3 c_5)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  - P(c_3)
P8,4 & A4,1      --> * R9:    + IN(c_3 c_5)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)
R4,1 & A19,2     --> * P10:   + IN(c_2 c_5)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  - P(c_2)
P10,4 & A3,1     --> * R11:   + IN(c_2 c_5)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)
A19,2 & A21,5    --> * R14:   All x,y:Any - P(y)  - H(f_1(y))  - P(x)
                                         - P(y)  - IN(x f_1(y))  + =(x y)
R14 4=1          --> * D15:   All x,y:Any - H(f_1(x))  - P(y)  - P(x)
                                         - IN(y f_1(x))  + =(y x)
D15,1 & A18,2    --> * R16:   All x,y:Any - P(x)  - P(y)  - IN(x f_1(y))
                                         + =(x y)  - P(y)
```

```
R16 2=5          --> * D17:   All x,y:Any - P(x)  - P(y)  - IN(x f_1(y))
                                           + =(x y)
R5,1 & D17,3     --> * P18:   All x:Any - IN(x c_5)  + =(f_1(c_1) c_6)
                                       + =(f_1(c_1) c_7)  - P(x)  - P(c_1)
                                       + =(x c_1)
P18,5 & A2,1     --> * R19:   All x:Any - IN(x c_5)  + =(f_1(c_1) c_6)
                                       + =(f_1(c_1) c_7)  - P(x)
                                       + =(x c_1)
R11,1 & R19,1    --> * R20:   + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_2)
                              + =(c_2 c_1)
R20,6 & A6,1     --> * R21:   + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_2)
R21,5 & A3,1     --> * R22:   + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)
R9,1 & R19,1     --> * R23:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_3)
                              + =(c_3 c_1)
R23,6 & A7,1     --> * R24:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_3)
R24,5 & A4,1     --> * R25:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)
R7,1 & R19,1     --> * R26:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_4)
                              + =(c_4 c_1)
R26,6 & A8,1     --> * R27:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)  - P(c_4)
R27,5 & A5,1     --> * R28:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_1) c_6)  + =(f_1(c_1) c_7)
R4,1 & D17,3     --> * P35:   All x:Any - IN(x c_5)  + =(f_1(c_2) c_6)
                                       + =(f_1(c_2) c_7)  - P(x)  - P(c_2)
                                       + =(x c_2)
P35,5 & A3,1     --> * R36:   All x:Any - IN(x c_5)  + =(f_1(c_2) c_6)
                                       + =(f_1(c_2) c_7)  - P(x)
                                       + =(x c_2)
R9,1 & R36,1     --> * R37:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_3)
                              + =(c_3 c_2)
R37,6 & A9,1     --> * R38:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_3)
R38,5 & A4,1     --> * R39:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
R7,1 & R36,1     --> * R40:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_4)
                              + =(c_4 c_2)
R40,6 & A10,1    --> * R41:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_4)
R41,5 & A5,1     --> * R42:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
R39,3 & A19,2    --> * P43:   + IN(c_2 c_6)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_2)
P43,5 & A3,1     --> * R44:   + IN(c_2 c_6)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
R42,3 & A19,2    --> * P45:   + IN(c_2 c_6)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)  - P(c_2)
P45,5 & A3,1     --> * R46:   + IN(c_2 c_6)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
R3,1 & D17,3     --> * P47:   All x:Any - IN(x c_5)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  - P(x)  - P(c_3)
                                       + =(x c_3)
P47,5 & A4,1     --> * R48:   All x:Any - IN(x c_5)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  - P(x)
                                       + =(x c_3)
R7,1 & R48,1     --> * R49:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)  - P(c_4)
```

```
                              + =(c_4 c_3)
R49,6 & A11,1    --> * R50:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)  - P(c_4)
R50,5 & A5,1     --> * R51:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
R51,3 & A19,2    --> * P52:   + IN(c_3 c_6)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)  - P(c_3)
P52,5 & A4,1     --> * R53:   + IN(c_3 c_6)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
R22,3 & D17,3    --> * P56:   All x:Any - IN(x c_6)  + =(f_1(c_2) c_6)
                                       + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  - P(c_1)  + =(x c_1)
P56,6 & A2,1     --> * R57:   All x:Any - IN(x c_6)  + =(f_1(c_2) c_6)
                                       + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  + =(x c_1)
R25,3 & D17,3    --> * P58:   All x:Any - IN(x c_6)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  - P(c_1)  + =(x c_1)
P58,6 & A2,1     --> * R59:   All x:Any - IN(x c_6)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  + =(x c_1)
R44,1 & R59,1    --> * R60:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
                              + =(c_2 c_1)
R60 2=5          --> * D61:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_1) c_7)  - P(c_2)  + =(c_2 c_1)
D61 1=4          --> * D62:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
                              + =(c_2 c_1)
D62,6 & A6,1     --> * R63:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
R63,5 & A3,1     --> * R64:   + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
R28,3 & D17,3    --> * P67:   All x:Any - IN(x c_6)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  - P(c_1)  + =(x c_1)
P67,6 & A2,1     --> * R68:   All x:Any - IN(x c_6)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_1) c_7)
                                       - P(x)  + =(x c_1)
R53,1 & R68,1    --> * R69:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_1) c_7)  - P(c_3)
                              + =(c_3 c_1)
R69 2=5          --> * D70:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_1) c_7)  - P(c_3)  + =(c_3 c_1)
D70 1=4          --> * D71:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)  - P(c_3)
                              + =(c_3 c_1)
D71,6 & A7,1     --> * R72:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)  - P(c_3)
R72,5 & A4,1     --> * R73:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)
R46,1 & R68,1    --> * R74:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
                              + =(c_2 c_1)
R74 2=5          --> * D75:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_1) c_7)  - P(c_2)  + =(c_2 c_1)
D75 1=4          --> * D76:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
                              + =(c_2 c_1)
```

```
D76,6 & A6,1     --> * R77:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_2)
R77,5 & A3,1     --> * R78:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
R78,4 & A19,2    --> * P81:   + IN(c_1 c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)  - P(c_1)
P81,5 & A2,1     --> * R82:   + IN(c_1 c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
R39,3 & D17,3    --> * P83:   All x:Any - IN(x c_6)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  - P(c_2)  + =(x c_2)
P83,6 & A3,1     --> * R84:   All x:Any - IN(x c_6)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  + =(x c_2)
R42,3 & D17,3    --> * P85:   All x:Any - IN(x c_6)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  - P(c_2)  + =(x c_2)
P85,6 & A3,1     --> * R86:   All x:Any - IN(x c_6)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  + =(x c_2)
R53,1 & R86,1    --> * R87:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)  - P(c_3)
                              + =(c_3 c_2)
R87 2=5          --> * D88:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_2) c_7)  - P(c_3)  + =(c_3 c_2)
D88 1=4          --> * D89:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_3)
                              + =(c_3 c_2)
D89,6 & A9,1     --> * R90:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_3)
R90,5 & A4,1     --> * R91:   + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
R91,4 & A19,2    --> * P92:   + IN(c_2 c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)  - P(c_2)
P92,5 & A3,1     --> * R93:   + IN(c_2 c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
R64,4 & D17,3    --> * P96:   All x:Any - IN(x c_7)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  - P(c_1)  + =(x c_1)
P96,6 & A2,1     --> * R97:   All x:Any - IN(x c_7)  + =(f_1(c_3) c_6)
                                       + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                                       - P(x)  + =(x c_1)
R73,4 & D17,3    --> * P98:   All x:Any - IN(x c_7)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
                                       - P(x)  - P(c_1)  + =(x c_1)
P98,6 & A2,1     --> * R99:   All x:Any - IN(x c_7)  + =(f_1(c_4) c_6)
                                       + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
                                       - P(x)  + =(x c_1)
R93,1 & R99,1    --> * R100:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_6)
                              + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)  - P(c_2)
                              + =(c_2 c_1)
R100 1=4         --> * D101:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  - P(c_2)  + =(c_2 c_1)
D101 2=4         --> * D102:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(f_1(c_3) c_7)  - P(c_2)
                              + =(c_2 c_1)
D102 3=4         --> * D103:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  - P(c_2)  + =(c_2 c_1)
D103,4 & A3,1    --> * R104:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  + =(c_2 c_1)
R104,4 & A6,1    --> * R105:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
```

```
                                + =(f_1(c_3) c_7)
R105,3 & A19,2   --> * P106:  + IN(c_3 c_7)  + =(f_1(c_4) c_6)
                                + =(f_1(c_4) c_7)  - P(c_3)
P106,4 & A4,1    --> * R107:  + IN(c_3 c_7)  + =(f_1(c_4) c_6)
                                + =(f_1(c_4) c_7)
R107,1 & A21,4   --> * R108:  All x:Any + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                         - H(c_7)  - P(c_3)  - P(x)
                                         - IN(x c_7)  + =(c_3 x)
R108,4 & A4,1    --> * R109:  All x:Any + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                         - H(c_7)  - P(x)  - IN(x c_7)
                                         + =(c_3 x)
R109,3 & A14,1   --> * R110:  All x:Any + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                         - P(x)  - IN(x c_7)  + =(c_3 x)
R82,1 & R110,4   --> * R111:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_2) c_7)  + =(f_1(c_4) c_6)
                                + =(f_1(c_4) c_7)  - P(c_1)  + =(c_3 c_1)
R111 1=4         --> * D112:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_2) c_7)  + =(f_1(c_4) c_7)  - P(c_1)
                                + =(c_3 c_1)
D112 2=4         --> * D113:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_2) c_7)  - P(c_1)  + =(c_3 c_1)
D113,4 & A2,1    --> * R114:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_2) c_7)  + =(c_3 c_1)
R114,4 & A7,1    --> * R115:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_2) c_7)
R115,3 & A19,2   --> * P116:  + IN(c_2 c_7)  + =(f_1(c_4) c_6)
                                + =(f_1(c_4) c_7)  - P(c_2)
P116,4 & A3,1    --> * R117:  + IN(c_2 c_7)  + =(f_1(c_4) c_6)
                                + =(f_1(c_4) c_7)
R117,1 & R110,4  --> * R118:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)  - P(c_2)
                                + =(c_3 c_2)
R118 1=3         --> * D119:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(f_1(c_4) c_7)  - P(c_2)  + =(c_3 c_2)
D119 2=3         --> * D120:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)  - P(c_2)
                                + =(c_3 c_2)
D120,3 & A3,1    --> * R121:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
                                + =(c_3 c_2)
R121,3 & A9,1    --> * R122:  + =(f_1(c_4) c_6)  + =(f_1(c_4) c_7)
R122,1 & A19,2   --> * P123:  + IN(c_4 c_6)  + =(f_1(c_4) c_7)  - P(c_4)
P123,3 & A5,1    --> * R124:  + IN(c_4 c_6)  + =(f_1(c_4) c_7)
R124,1 & R84,1   --> * R125:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_4)
                                + =(c_4 c_2)
R125,6 & A10,1   --> * R126:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_4)
R126,5 & A5,1    --> * R127:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
R124,1 & R59,1   --> * R128:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)  - P(c_4)
                                + =(c_4 c_1)
R128,6 & A8,1    --> * R129:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)  - P(c_4)
R129,5 & A5,1    --> * R130:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                                + =(f_1(c_3) c_7)  + =(f_1(c_1) c_7)
R124,1 & R57,1   --> * R131:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                                + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_4)
                                + =(c_4 c_1)
R131,6 & A8,1    --> * R132:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                                + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_4)
R132,5 & A5,1    --> * R133:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                                + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
R122,1 & D17,3   --> * P134:  All x:Any - IN(x c_6)  + =(f_1(c_4) c_7)
                                         - P(x)  - P(c_4)  + =(x c_4)
P134,4 & A5,1    --> * R135:  All x:Any - IN(x c_6)  + =(f_1(c_4) c_7)
```

```
                                        - P(x)  + =(x c_4)
R130,4 & A19,2   --> * P138:  + IN(c_1 c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)  - P(c_1)
P138,5 & A2,1    --> * R139:  + IN(c_1 c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
R133,4 & A19,2   --> * P140:  + IN(c_1 c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_1)
P140,5 & A2,1    --> * R141:  + IN(c_1 c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
R127,4 & D17,3   --> * P142:  All x:Any - IN(x c_7)  + =(f_1(c_4) c_7)
                                        + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                                        - P(x)  - P(c_2)  + =(x c_2)
P142,6 & A3,1    --> * R143:  All x:Any - IN(x c_7)  + =(f_1(c_4) c_7)
                                        + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                                        - P(x)  + =(x c_2)
R139,1 & R143,1  --> * R144:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)  - P(c_1)
                              + =(c_1 c_2)
R144 1=4         --> * D145:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  - P(c_1)  + =(c_1 c_2)
D145 2=4         --> * D146:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(f_1(c_3) c_7)  - P(c_1)
                              + =(c_1 c_2)
D146 3=4         --> * D147:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  - P(c_1)  + =(c_1 c_2)
D147,4 & A2,1    --> * R148:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  + =(c_1 c_2)
R148,4 & A6,1    --> * R149:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)
R149,2 & A19,2   --> * P150:  + IN(c_3 c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)  - P(c_3)
P150,4 & A4,1    --> * R151:  + IN(c_3 c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_3) c_7)
R151,1 & R135,1  --> * R152:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
                              + =(f_1(c_4) c_7)  - P(c_3)  + =(c_3 c_4)
R152 1=3         --> * D153:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)  - P(c_3)
                              + =(c_3 c_4)
D153,3 & A4,1    --> * R154:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
                              + =(c_3 c_4)
R154,3 & A11,1   --> * R155:  + =(f_1(c_4) c_7)  + =(f_1(c_3) c_7)
R155,2 & D17,3   --> * P158:  All x:Any - IN(x c_7)  + =(f_1(c_4) c_7)
                                        - P(x)  - P(c_3)  + =(x c_3)
P158,4 & A4,1    --> * R159:  All x:Any - IN(x c_7)  + =(f_1(c_4) c_7)
                                        - P(x)  + =(x c_3)
R141,1 & R159,1  --> * R160:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(f_1(c_4) c_7)  - P(c_1)
                              + =(c_1 c_3)
R160 1=4         --> * D161:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  - P(c_1)  + =(c_1 c_3)
D161,4 & A2,1    --> * R162:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(c_1 c_3)
R162,4 & A7,1    --> * R163:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)
R163,2 & A19,2   --> * P164:  + IN(c_2 c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)  - P(c_2)
P164,4 & A3,1    --> * R165:  + IN(c_2 c_6)  + =(f_1(c_4) c_7)
                              + =(f_1(c_2) c_7)
R165,1 & R135,1  --> * R166:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
                              + =(f_1(c_4) c_7)  - P(c_2)  + =(c_2 c_4)
R166 1=3         --> * D167:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)  - P(c_2)
                              + =(c_2 c_4)
D167,3 & A3,1    --> * R168:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
                              + =(c_2 c_4)
```

```
R168,3 & A10,1   --> * R169:  + =(f_1(c_4) c_7)  + =(f_1(c_2) c_7)
R169,2 & A19,2   --> * P170:  + IN(c_2 c_7)  + =(f_1(c_4) c_7)  - P(c_2)
P170,3 & A3,1    --> * R171:  + IN(c_2 c_7)  + =(f_1(c_4) c_7)
R171,1 & R159,1  --> * R172:  + =(f_1(c_4) c_7)  + =(f_1(c_4) c_7)  - P(c_2)
                              + =(c_2 c_3)
R172 1=2         --> * D173:  + =(f_1(c_4) c_7)  - P(c_2)  + =(c_2 c_3)
D173,2 & A3,1    --> * R174:  + =(f_1(c_4) c_7)  + =(c_2 c_3)
R174,2 & A9,1    --> * R175:  + =(f_1(c_4) c_7)
R175,1 & A19,2   --> * P176:  + IN(c_4 c_7)  - P(c_4)
P176,2 & A5,1    --> * R177:  + IN(c_4 c_7)
R97,1 & R177,1   --> * R178:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  - P(c_4)  + =(c_4 c_1)
R178,4 & A5,1    --> * R179:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  + =(c_4 c_1)
R179,4 & A8,1    --> * R180:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)
R175,1 & D17,3   --> * P181:  All x:Any - IN(x c_7)  - P(x)  - P(c_4)
                                        + =(x c_4)
P181,3 & A5,1    --> * R182:  All x:Any - IN(x c_7)  - P(x)  + =(x c_4)
R180,3 & A19,2   --> * P183:  + IN(c_2 c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)  - P(c_2)
P183,4 & A3,1    --> * R184:  + IN(c_2 c_7)  + =(f_1(c_3) c_6)
                              + =(f_1(c_3) c_7)
R184,1 & R182,1  --> * R185:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)  - P(c_2)
                              + =(c_2 c_4)
R185,3 & A3,1    --> * R186:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
                              + =(c_2 c_4)
R186,3 & A10,1   --> * R187:  + =(f_1(c_3) c_6)  + =(f_1(c_3) c_7)
R187,1 & A19,2   --> * P188:  + IN(c_3 c_6)  + =(f_1(c_3) c_7)  - P(c_3)
P188,3 & A4,1    --> * R189:  + IN(c_3 c_6)  + =(f_1(c_3) c_7)
R189,1 & R57,1   --> * R190:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_3)
                              + =(c_3 c_1)
R190,6 & A7,1    --> * R191:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)  - P(c_3)
R191,5 & A4,1    --> * R192:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(f_1(c_1) c_7)
R187,1 & D17,3   --> * P193:  All x:Any - IN(x c_6)  + =(f_1(c_3) c_7)
                                       - P(x)  - P(c_3)  + =(x c_3)
P193,4 & A4,1    --> * R194:  All x:Any - IN(x c_6)  + =(f_1(c_3) c_7)
                                       - P(x)  + =(x c_3)
R192,4 & A19,2   --> * P195:  + IN(c_1 c_7)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)  - P(c_1)
P195,5 & A2,1    --> * R196:  + IN(c_1 c_7)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_6)  + =(f_1(c_2) c_7)
R196,1 & R182,1  --> * R197:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  - P(c_1)  + =(c_1 c_4)
R197,4 & A2,1    --> * R198:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)  + =(c_1 c_4)
R198,4 & A8,1    --> * R199:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_6)
                              + =(f_1(c_2) c_7)
R199,2 & A19,2   --> * P200:  + IN(c_2 c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)  - P(c_2)
P200,4 & A3,1    --> * R201:  + IN(c_2 c_6)  + =(f_1(c_3) c_7)
                              + =(f_1(c_2) c_7)
R201,1 & R194,1  --> * R202:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                              + =(f_1(c_3) c_7)  - P(c_2)  + =(c_2 c_3)
R202 1=3         --> * D203:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)  - P(c_2)
                              + =(c_2 c_3)
D203,3 & A3,1    --> * R204:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
                              + =(c_2 c_3)
R204,3 & A9,1    --> * R205:  + =(f_1(c_3) c_7)  + =(f_1(c_2) c_7)
R205,2 & A19,2   --> * P206:  + IN(c_2 c_7)  + =(f_1(c_3) c_7)  - P(c_2)
P206,3 & A3,1    --> * R207:  + IN(c_2 c_7)  + =(f_1(c_3) c_7)
R207,1 & R182,1  --> * R208:  + =(f_1(c_3) c_7)  - P(c_2)  + =(c_2 c_4)
```

```
R208,2 & A3,1    --> * R209:  + =(f_1(c_3) c_7)  + =(c_2 c_4)
R209,2 & A10,1   --> * R210:  + =(f_1(c_3) c_7)
R210,1 & A19,2   --> * P211:  + IN(c_3 c_7)  - P(c_3)
P211,2 & A4,1    --> * R212:  + IN(c_3 c_7)
R212,1 & R182,1  --> * R213:  - P(c_3)  + =(c_3 c_4)
R213,1 & A4,1    --> * R214:  + =(c_3 c_4)
R214,1 & A11,1   --> * R215:  []
```

B.3 Other Examples

The first two examples of this section are taken from the theory of zero divisor free rings and show the usefulness of narrowing to find E-unifiers for arbitrary theories. They are not presented in the version originally computed on the machine because we used a different system to run them. Especially for the second example the MKRP-system was not able to find a solution without the help of a user, because of various implementation restrictions in this almost ancient system. The automatically generated version computed by a program with fully integrated narrowing is published by J. Richts [Ric91].

B.3.1 Cancellation

The cancellation law for multiplication is valid in a zero divisor free ring.

```
Formulae given to the editor
============================

Axioms:   * Ring *
          ALL X,Y,Z +(+(X Y) Z) = +(X +(Y Z))
          ALL X +(0 X) = X
          ALL X +(-(X) X) = 0
          ALL X,Y,Z *(*(X Y) Z) = *(X *(Y Z))
          ALL X,Y,Z *(X +(Y Z)) = +(*(X Y) *(X Z))
          ALL X,Y,Z *(+(Y Z) X) = +(*(Y X) *(Z X))
          * With One *
          ALL X *(1 X) = X
          ALL X *(X 1) = X
          * Zero Divisor Free *
          ALL X,Y *(X Y) = 0 IMPL X = 0 OR Y = 0

Theorems: * Cancellation *
          ALL X,Y,Z *(X Y) = *(Z Y) AND NOT (Y = 0) IMPL X = Z

Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)
        * A5:   All x,y,z:Any + =(*(*(z y) x) *(z *(y x)))
        * A6:   All x,y,z:Any + =(*(z +(y x)) +(*(z y) *(z x)))
        * A7:   All x,y,z:Any + =(*(+(z y) x) +(*(z x) *(y x)))
        * A8:   All x:Any + =(*(1 x) x)
        * A9:   All x:Any + =(*(x 1) x)
        * A10:  All x,y:Any - =(*(y x) 0)  + =(y 0)  + =(x 0)


Set of Theorem Clauses Resulting from Normalization
===================================================

        * T11: + =(*(c_2 c_1) *(c_3 c_1))
        * T12: - =(c_1 0)
```

```
       * T13: - =(c_2 c_3)

Refutation:
===========

A4,1 & A2,1      --> * P1:   All x,y:Any + =(+(0 y) +(-(x) +(x y)))
P1,1 & A3        --> * RW2:  All x,y:Any + =(y +(-(x) +(x y)))
A4,1 & RW2,1     --> * P4:   All x:Any + =(x +(-(-(x)) 0))
P4,1 & RW2,1     --> * P10:  All x:Any + =(0 +(-(-(-(x))) x))
P10,1 & RW2,1    --> * P11:  All x:Any + =(x +(-(-(-(-(x)))) 0))
P11,1 & P4       --> * RW12: All x:Any + =(x -(-(x)))
P4,1 & RW12      --> * RW14: All x:Any + =(x +(x 0))
RW12,1 & A4,1    --> * P15:  All x:Any + =(+(x -(x)) 0)
A3,1 & A6,1      --> * P16:  All x,y:Any + =(*(y x) +(*(y 0) *(y x)))
A9,1 & P16,1     --> * P17:  All x:Any + =(*(x 1) +(*(x 0) x))
P17,1 & A9       --> * RW18: All x:Any + =(x +(*(x 0) x))
RW14,1 & RW18,1  --> * P19:  + =(0 *(0 0))
A3,1 & A7,1      --> * P21:  All x,y:Any + =(*(y x) +(*(0 x) *(y x)))
A8,1 & P21,1     --> * P22:  All x:Any + =(*(1 x) +(*(0 x) x))
P22,1 & A8       --> * RW23: All x:Any + =(x +(*(0 x) x))
RW23,1 & RW2,1   --> * P24:  All x:Any + =(x +(-(*(0 x)) x))
P19,1 & A5,1     --> * P28:  All x:Any + =(*(0 x) *(0 *(0 x)))
P28,1 & P24,1    --> * P29:  All x:Any + =(*(0 x) +(-(*(0 x)) *(0 x)))
P29,1 & A4       --> * RW30: All x:Any + =(*(0 x) 0)
P28,1 & RW30     --> * RW31: All x:Any + =(*(0 x) *(0 0))
RW31,1 & RW30    --> * RW32: All x:Any + =(*(0 x) 0)
A4,1 & A7,1      --> * P82:  All x,y:Any + =(*(0 y) +(*(-(x) y) *(x y)))
P82,1 & RW32     --> * RW83: All x,y:Any + =(0 +(*(-(y) x) *(y x)))
RW83,1 & RW2,1   --> * P87:  All x,y:Any + =(*(y x) +(-(*(-(y) x)) 0))
P87,1 & RW14     --> * RW88: All x,y:Any + =(*(y x) -(*(-(y) x)))
RW88,1 & A4,1    --> * P97:  All x,y:Any + =(+(*(y x) *(-(y) x)) 0)
P97,1 & RW2,1    --> * P98:  All x,y:Any + =(*(-(y) x) +(-(*(y x)) 0))
P98,1 & RW14     --> * RW99: All x,y:Any + =(*(-(y) x) -(*(y x)))
A7,1 & A10,1     --> * P154: All x,y,z:Any - =(+(*(z y) *(x y)) 0)
                                           + =(+(z x) 0)  + =(y 0)
T11,1 & P154,1   --> * P155: All x:Any - =(+(*(c_2 c_1) *(x c_1)) 0)
                                       + =(+(c_3 x) 0)  + =(c_1 0)
P155,3 & T12,1   --> * R156: All x:Any - =(+(*(c_2 c_1) *(x c_1)) 0)
                                       + =(+(c_3 x) 0)
RW99,1 & R156,1  --> * P167: All x:Any - =(+(*(c_2 c_1) -(*(x c_1))) 0)
                                       + =(+(c_3 -(x)) 0)
P15,1 & P167,1   --> * P168: - =(0 0)  + =(+(c_3 -(c_2)) 0)
P168,1 & A1,1    --> * R169: + =(+(c_3 -(c_2)) 0)
R169,1 & RW2,1   --> * P170: + =(-(c_2) +(-(c_3) 0))
P170,1 & RW14    --> * RW171:+ =(-(c_2) -(c_3))
RW171,1 & RW12,1 --> * P172: + =(c_3 -(-(c_2)))
P172,1 & RW12    --> * RW173:+ =(c_3 c_2)
RW173,1 & T13,1  --> * R174: []
```

B.3.2 Two Square Roots

There are exactly two square roots of 1 in a zero divisor free ring, namely -1 and 1.

```
Formulae given to the editor
============================

Axioms:   * Ring *
          ALL X,Y,Z +(+(X Y) Z) = +(X +(Y Z))
          ALL X +(0 X) = X
          ALL X +(-(X) X) = 0
          ALL X,Y,Z *(*(X Y) Z) = *(X *(Y Z))
          ALL X,Y,Z *(X +(Y Z)) = +(*(X Y) *(X Z))
          ALL X,Y,Z *(+(Y Z) X) = +(*(Y X) *(Z X))
          * With One *
```

```
          ALL X *(1 X) = X
          ALL X *(X 1) = X
          * Zero Divisor Free *
          ALL X,Y *(X Y) = 0 IMPL X = 0 OR Y = 0

Theorems: * Two Square Roots *
          ALL X +(*(X X) -(1)) = 0 IMPL X = 1 OR X = -(1)

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)
          A5:   All x,y,z:Any + =(*(*(z y) x) *(z *(y x)))
        * A6:   All x,y,z:Any + =(*(z +(y x)) +(*(z y) *(z x)))
        * A7:   All x,y,z:Any + =(*(+(z y) x) +(*(z x) *(y x)))
        * A8:   All x:Any + =(*(1 x) x)
        * A9:   All x:Any + =(*(x 1) x)
        * A10:  All x,y:Any - =(*(y x) 0)  + =(y 0)  + =(x 0)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T11: + =(+(*(c_1 c_1) -(1)) 0)
        * T12: - =(c_1 1)
        * T13: - =(c_1 -(1))

Refutation:
===========

A4,1 & A2,1      --> * P1:   All x,y:Any + =(+(0 y) +(-(x) +(x y)))
P1,1 & A3        --> * RW2:  All x,y:Any + =(y +(-(x) +(x y)))
A4,1 & RW2,1     --> * P4:   All x:Any + =(x +(-(-(x)) 0))
T11,1 & A2,1     --> * P10:  All x:Any + =(+(0 x) +(*(c_1 c_1)
                                                   +(-(1) x)))
P10,1 & A3       --> * RW11: All x:Any + =(x +(*(c_1 c_1) +(-(1) x)))
A4,1 & RW11,1    --> * P12:  + =(1 +(*(c_1 c_1) 0))
RW2,1 & RW11,1   --> * P13:  All x:Any + =(+(1 x) +(*(c_1 c_1) x))
P12,1 & P13      --> * RW16: + =(1 +(1 0))
RW16,1 & A6,1    --> * P17:  All x:Any + =(*(x 1) +(*(x 1) *(x 0)))
P17,1 & A9       --> * RW18: All x:Any + =(*(x 1) +(x *(x 0)))
RW18,1 & A9      --> * RW19: All x:Any + =(x +(x *(x 0)))
RW19,1 & RW2,1   --> * P21:  All x:Any + =(*(x 0) +(-(x) x))
P21,1 & A4       --> * RW22: All x:Any + =(*(x 0) 0)
RW19,1 & RW22    --> * RW23: All x:Any + =(x +(x 0))
P4,1 & RW23      --> * RW24: All x:Any + =(x -(-(x)))
P13,1 & RW23,1   --> * P25:  + =(*(c_1 c_1) +(1 0))
P25,1 & RW23     --> * RW26: + =(*(c_1 c_1) 1)
RW24,1 & A4,1    --> * P28:  All x:Any + =(+(x -(x)) 0)
A4,1 & A6,1      --> * P32:  All x,y:Any + =(*(y 0) +(*(y -(x)) *(y x)))
P32,1 & RW22     --> * RW33: All x,y:Any + =(0 +(*(y -(x)) *(y x)))
RW33,1 & RW2,1   --> * P42:  All x,y:Any + =(*(y x) +(-(*(y -(x))) 0))
P42,1 & RW23     --> * RW43: All x,y:Any + =(*(y x) -(*(y -(x))))
RW24,1 & RW43,1  --> * P53:  All x,y:Any + =(*(y -(x)) -(*(y x)))
A7,1 & A10,1     --> * P235: All x,y,z:Any - =(+(*(z y) *(x y)) 0)
                                           + =(+(z x) 0)  + =(y 0)
A8,1 & P235,1    --> * P236: All x,y:Any - =(+(y *(x y)) 0)
                                         + =(+(1 x) 0)  + =(y 0)
A2,1 & P236,1    --> * P237: All x,y,z:Any - =(+(z +(y *(x +(z y)))) 0)
                                           + =(+(1 x) 0)  + =(+(z y) 0)
P237,1 & A6      --> * RW238:All x,y,z:Any - =(+(z +(y +(*(x z) *(x y))))
                                              0)
                                           + =(+(1 x) 0)  + =(+(z y) 0)
```

```
A9,1 & RW238,1   --> * P239: All x,y:Any - =(+(1 +(y +(x *(x y)))) 0)
                                          + =(+(1 x) 0)  + =(+(1 y) 0)
RW2,1 & P239,1   --> * P240: All x:Any - =(+(1 *(x -(x))) 0)
                                       + =(+(1 x) 0)  + =(+(1 -(x)) 0)
P240,1 & P53     --> * RW241:All x:Any - =(+(1 -(*(x x))) 0)
                                       + =(+(1 x) 0)  + =(+(1 -(x)) 0)
RW26,1 & RW241,1 --> * P242: - =(+(1 -(1)) 0)  + =(+(1 c_1) 0)
                           + =(+(1 -(c_1)) 0)
P242,1 & P28,1   --> * R243: + =(+(1 c_1) 0)  + =(+(1 -(c_1)) 0)
R243,2 & A2,1    --> * P244: All x:Any + =(+(0 x) +(1 +(-(c_1) x)))
                                       + =(+(1 c_1) 0)
P244,1 & A3      --> * RW245:All x:Any + =(x +(1 +(-(c_1) x)))
                                       + =(+(1 c_1) 0)
P28,1 & RW245,1  --> * P246: + =(-(-(c_1)) +(1 0))  + =(+(1 c_1) 0)
P246,1 & RW23    --> * RW247:+ =(-(-(c_1)) 1)  + =(+(1 c_1) 0)
RW247,1 & RW24   --> * RW248:+ =(c_1 1)  + =(+(1 c_1) 0)
RW248,1 & T12,1  --> * R249: + =(+(1 c_1) 0)
R249,1 & RW2,1   --> * P250: + =(c_1 +(-(1) 0))
P250,1 & RW23    --> * RW251:+ =(c_1 -(1))
RW251,1 & T13,1  --> * R252: []
```

B.3.3 Commutator

```
Formulae given to the editor
============================

Axioms:   ALL X,Y,Z A(A(A(S X) Y) Z) = A(A(X Z) A(Y Z))
          ALL X,Y A(A(K X) Y) = X

Theorems: EX U ALL X,Y A(A(U X) Y) = A(Y A(A(X X) Y))

Refutation:
===========

Initial Clauses:  * A2:   All x,y,z:Any + =(a(a(a(s z) y) x)
                                           a(a(z x) a(y x)))
                  * A3:   All x,y:Any + =(a(a(k y) x) y)
                  * T4:   All x:Any - =(a(a(x f_2(x)) f_1(x))
                                        a(f_1(x) a(a(f_2(x) f_2(x))
                                                  f_1(x))))

A3,1 & A2,1   --> * P1:   All x,y:Any + =(a(a(a(s k) y) x) x)
A3,1 & A2,1   --> * P3:   All x,y,z:Any + =(a(a(a(s a(k z)) y) x)
                                           a(z a(y x)))
P1,1 & A2,1   --> * P6:   All x,y,z:Any + =(a(a(a(s a(a(s k) z)) y) x)
                                           a(x a(y x)))
P1,1 & A2,1   --> * P9:   All x,y,z:Any + =(a(a(a(s z) a(a(s k) y)) x)
                                           a(a(z x) x))
P9,1 & P3,1   --> * P104: All x,y,z:Any + =(a(a(a(s a(s a(k z)))
                                                a(a(s k) y))
                                              x)
                                           a(z a(x x)))
P104,1 & P6,1 --> * P105: All x,y,z,u:Any
                           + =(a(a(a(a(s a(s a(k a(s a(a(s k) u)))))
                                     a(a(s k) z)) y) x)
                              a(x a(a(y y) x)))
P105,1 & T4,1 --> * R106: []
```

B.3.4 Group Completion

We prove $-(x+y) = (-y)+(-x)$ for a group just to present the example where all steps must be performed to derive a canonical rewrite system for groups.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(+(+(z y) x) +(z +(y x)))
        * A3:   All x:Any + =(+(0 x) x)
        * A4:   All x:Any + =(+(-(x) x) 0)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T5: - =(+(-(c_1) -(c_2)) -(+(c_2 c_1)))


Refutation:
===========

A4,1 & A2,1     --> * P1:    All x,y:Any + =(+(0 y) +(-(x) +(x y)))
P1,1 & A3       --> * RW2:   All x,y:Any + =(y +(-(x) +(x y)))
A4,1 & RW2,1    --> * P4:    All x:Any + =(x +(-(-(x)) 0))
P4,1 & RW2,1    --> * P12:   All x:Any + =(0 +(-(-(-(x))) x))
P12,1 & RW2,1   --> * P13:   All x:Any + =(x +(-(-(-(-(x)))) 0))
P13,1 & P4      --> * RW14:  All x:Any + =(x -(-(x)))
P4,1 & RW14     --> * RW15:  All x:Any + =(x +(x 0))
RW14,1 & A4,1   --> * P18:   All x:Any + =(+(x -(x)) 0)
P18,1 & A2,1    --> * P20:   All x,y:Any + =(0 +(y +(x -(+(y x)))))
P20,1 & RW2,1   --> * P21:   All x,y:Any + =(+(y -(+(x y))) +(-(x) 0))
P21,1 & RW15    --> * RW22:  All x,y:Any + =(+(y -(+(x y))) -(x))
RW22,1 & RW2,1  --> * P27:   All x,y:Any + =(-(+(y x)) +(-(x) -(y)))
P27,1 & T5,1    --> * R28:   []
```

B.3.5 Central Groupoid

This example can be found in [KB70] or [SK90, Example 3.5].

```
Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   All x,y,z:Any + =(*(*(z y) *(y x)) y)
        * A3:   All x:Any + =(*(*(x x) x) f(x))
        * A4:   All x:Any + =(*(x *(x x)) g(x))
        * A5:   All x,y:Any + =(*(g(y) x) *(y x))

Set of Theorem Clauses Resulting from Normalization
===================================================

* T6: - =(*(*(c_1 c_2) c_3) *(f(c_2) c_3))

Refutation:
===========

A2,1 & A4,1     --> * P3:    All x:Any + =(*(*(x x) x) g(*(x x)))
P3,1 & A3       --> * RW4:   All x:Any + =(f(x) g(*(x x)))
A3,1 & A2,1     --> * P5:    All x,y:Any + =(*(f(y) *(y x)) y)
A4,1 & P5,1     --> * P6:    All x:Any + =(*(f(x) g(x)) x)
A4,1 & A2,1     --> * P7:    All x,y:Any + =(*(*(y x) g(x)) x)
P6,1 & A2,1     --> * P11:   All x,y:Any + =(*(y *(g(y) x)) g(y))
P11,1 & A5      --> * RW12:  All x,y:Any + =(*(y *(y x)) g(y))
RW12,1 & A5,1   --> * P13:   All x,y:Any + =(g(g(y)) *(y *(g(y) x)))
P13,1 & A5      --> * RW14:  All x,y:Any + =(g(g(y)) *(y *(y x)))
RW14,1 & RW12   --> * RW15:  All x:Any + =(g(g(x)) g(x))
P7,1 & RW12,1   --> * P46:   All x,y:Any + =(*(*(y x) x) g(*(y x)))
RW15,1 & P7,1   --> * P47:   All x,y:Any + =(*(*(y g(x)) g(x)) g(x))
P47,1 & P46     --> * RW48:  All x,y:Any + =(g(*(y g(x))) g(x))
```

```
RW48,1 & A5,1   --> * P51:   All x,y,z:Any + =(*(g(z) y) *(*(x g(z)) y))
P51,1 & A5      --> * RW52:  All x,y,z:Any + =(*(z y) *(*(x g(z)) y))
P5,1 & A2,1     --> * P53:   All x,y,z:Any + =(*(z *(*(z y) x)) *(z y))
RW52,1 & P53,1  --> * P54:   All x,y,z:Any + =(*(z *(y x)) *(z g(y)))
P54,1 & RW4,1   --> * P66:   All x,y:Any + =(f(*(y x)) g(*(*(y x) g(y))))
P66,1 & RW48    --> * RW67:  All x,y:Any + =(f(*(y x)) g(y))
P7,1 & RW67,1   --> * P70:   All x,y:Any + =(f(y) g(*(x y)))
P70,1 & A5,1    --> * P85:   All x,y,z:Any + =(*(f(z) y) *(*(x z) y))
P85,1 & T6,1    --> * R86:   []
```

B.3.6 Z22

Z22 is a finite cyclic group containing five elements (`a b c d e`) and their inverses (`a1 b1 c1 d1 e1`). The dependencies are expressed by the axioms `A2` to `A6`.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

        * A1:   All x:Any + =(x x)
        * A2:   All x:Any + =(a(b(c(x))) d(x))
        * A3:   All x:Any + =(b(c(d(x))) e(x))
        * A4:   All x:Any + =(c(d(e(x))) a(x))
        * A5:   All x:Any + =(d(e(a(x))) b(x))
        * A6:   All x:Any + =(e(a(b(x))) c(x))
        * A7:   All x:Any + =(a(a1(x)) x)
          A8:   All x:Any + =(a1(a(x)) x)
          A9:   All x:Any + =(b(b1(x)) x)
        * A10:  All x:Any + =(b1(b(x)) x)
        * A11:  All x:Any + =(c(c1(x)) x)
        * A12:  All x:Any + =(c1(c(x)) x)
        * A13:  All x:Any + =(d(d1(x)) x)
        * A14:  All x:Any + =(d1(d(x)) x)
        * A15:  All x:Any + =(e(e1(x)) x)
        * A16:  All x:Any + =(e1(e(x)) x)

Initial Operations on Axioms
============================

A2,1 & A4   --> * RW1:  All x:Any + =(c(d(e(b(c(x))))) d(x))
A5,1 & A4   --> * RW2:  All x:Any + =(d(e(c(d(e(x))))) b(x))
RW1,1 & RW2 --> * RW3:  All x:Any + =(c(d(e(d(e(c(d(e(c(x))))))))) d(x))
A3,1 & RW2  --> * RW4:  All x:Any + =(d(e(c(d(e(c(d(x))))))) e(x))
A6,1 & RW2  --> * RW5:  All x:Any + =(e(a(d(e(c(d(e(x))))))) c(x))
RW5,1 & A4  --> * RW6:  All x:Any + =(e(c(d(e(d(e(c(d(e(x))))))))) c(x))
A7,1 & A4   --> * RW7:  All x:Any + =(c(d(e(a1(x)))) x)
A10,1 & RW2 --> * RW10: All x:Any + =(b1(d(e(c(d(e(x)))))) x)

Set of Theorem Clauses Resulting from Normalization
===================================================

        * T17: - =(d(c_1) a(a(a(c_1))))  - =(d(c_2) e(e(e(e(e(c_2))))))

Initial Operations on Theorems
==============================

T17,1 & A4   --> * RW11: - =(d(c_1) c(d(e(a(a(c_1)))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW11,1 & A4  --> * RW12: - =(d(c_1) c(d(e(c(d(e(a(c_1))))))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW12,1 & A4  --> * RW13: - =(d(c_1) c(d(e(c(d(e(c(d(e(c_1))))))))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW13,1 & RW4 --> * RW14: - =(d(c_1) c(e(e(c_1))))  - =(d(c_2) e(e(e(e(e(c_2))))))

Refutation:
===========
```

```
RW3,1 & RW6,1      --> * P15:    All x:Any + =(e(d(x)) c(c(x)))
RW4,1 & RW10,1     --> * P16:    All x:Any + =(b1(e(x)) c(d(x)))
RW7,1 & RW10,1     --> * P17:    All x:Any + =(b1(d(e(x))) a1(x))
RW7,1 & P17        --> * RW18:   All x:Any + =(c(d(e(b1(d(e(x)))))) x)
A15,1 & RW18,1     --> * P21:    All x:Any + =(c(d(e(b1(d(x))))) e1(x))
P21,1 & A12,1      --> * P24:    All x:Any + =(c1(e1(x)) d(e(b1(d(x)))))
P24,1 & A14,1      --> * P27:    All x:Any + =(d1(c1(e1(x))) e(b1(d(x))))
P27,1 & A16,1      --> * P30:    All x:Any + =(e1(d1(c1(e1(x)))) b1(d(x)))
A13,1 & P30,1      --> * P40:    All x:Any + =(e1(d1(c1(e1(d1(x))))) b1(x))
P16,1 & P40        --> * RW47:   All x:Any + =(e1(d1(c1(e1(d1(e(x)))))) c(d(x)))
P15,1 & A12,1      --> * P50:    All x:Any + =(c1(e(d(x))) c(x))
A11,1 & P15,1      --> * P51:    All x:Any + =(e(d(c1(x))) c(x))
A13,1 & P50,1      --> * P52:    All x:Any + =(c1(e(x)) c(d1(x)))
P51,1 & A16,1      --> * P55:    All x:Any + =(e1(c(x)) d(c1(x)))
A15,1 & P52,1      --> * P58:    All x:Any + =(c1(x) c(d1(e1(x))))
RW47,1 & P58       --> * RW64:   All x:Any + =(e1(d1(c(d1(e1(e1(d1(e(x)))))))) c(d(x)))
P55,1 & P58        --> * RW69:   All x:Any + =(e1(c(x)) d(c(d1(e1(x)))))
A16,1 & RW69,1     --> * P70:    All x:Any + =(e1(c(e(x))) d(c(d1(x))))
P70,1 & A14,1      --> * P73:    All x:Any + =(d1(e1(c(e(x)))) c(d1(x)))
RW64,1 & P73       --> * RW79:   All x:Any + =(e1(d1(d1(e1(c(e(e1(e1(d1(e(x))))))))))) c(d(x)))
RW79,1 & A15       --> * RW80:   All x:Any + =(e1(d1(d1(e1(c(e1(d1(e(x)))))))) c(d(x)))
RW80,1 & A15,1     --> * P96:    All x:Any + =(e(c(d(x))) d1(d1(e1(c(e1(d1(e(x))))))))
P96,1 & A13,1      --> * P99:    All x:Any + =(d(e(c(d(x)))) d1(e1(c(e1(d1(e(x)))))))
P99,1 & P73,1      --> * P102:   All x:Any + =(d1(e1(c(e(e1(c(e1(d1(e(x))))))))) c(d(e(c(d(x))))))
P102,1 & A15       --> * RW103:  All x:Any + =(d1(e1(c(c(e1(d1(e(x)))))))) c(d(e(c(d(x))))))
RW103,1 & P15      --> * RW104:  All x:Any + =(d1(e1(e(d(e1(d1(e(x)))))))) c(d(e(c(d(x))))))
RW104,1 & A16      --> * RW105:  All x:Any + =(d1(d(e1(d1(e(x))))) c(d(e(c(d(x))))))
RW105,1 & A14      --> * RW106:  All x:Any + =(e1(d1(e(x))) c(d(e(c(d(x))))))
RW106,1 & RW3,1    --> * P110:   All x:Any + =(c(d(e(d(e(e1(d1(e(x))))))))) d(d(x)))
P110,1 & A15       --> * RW111:  All x:Any + =(c(d(e(d(d1(e(x)))))) d(d(x)))
RW111,1 & A13      --> * RW112:  All x:Any + =(c(d(e(e(x)))) d(d(x)))
A15,1 & RW112,1    --> * P113:   All x:Any + =(c(d(e(x))) d(d(e1(x))))
RW3,1 & P113       --> * RW114:  All x:Any + =(d(d(e1(d(e(c(d(e(c(x))))))))) d(x))
RW114,1 & P113     --> * RW115:  All x:Any + =(d(d(e1(d(e(d(d(e1(c(x))))))))) d(x))
RW6,1 & P113       --> * RW118:  All x:Any + =(e(d(d(e1(d(e(c(d(e(x))))))))) c(x))
RW118,1 & P113     --> * RW119:  All x:Any + =(e(d(d(e1(d(e(d(d(e1(x))))))))) c(x))
RW106,1 & P113     --> * RW120:  All x:Any + =(e1(d1(e(x))) d(d(e1(c(d(x))))))
RW115,1 & RW119    --> * RW123:
  All x:Any + =(d(d(e1(d(e(d(d(e1(e(d(d(e1(d(e(d(d(e1(x))))))))))))))))) d(x))
RW123,1 & A16      --> * RW124:  All x:Any + =(d(d(e1(d(e(d(d(d(d(d(e1(d(e(d(d(e1(x))))))))))))))))) d(x))
RW14,1 & RW119     --> * RW125:  - =(d(c_1) e(d(d(e1(d(e(d(d(e1(e(e(c_1))))))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW125,1 & A16      --> * RW126:  - =(d(c_1) e(d(d(e1(d(e(d(d(d(e(c_1))))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
P73,1 & RW119      --> * RW141:  All x:Any + =(d1(e1(e(d(d(e1(d(e(d(d(e1(e(x))))))))))))) c(d1(x)))
RW141,1 & RW119    --> * RW142:  All x:Any + =(d1(e1(e(d(d(e1(d(e(d(d(e1(e(x))))))))))))
                                             e(d(d(e1(d(e(d(d(e1(d1(x))))))))))
RW142,1 & A16      --> * RW143:  All x:Any + =(d1(d(d(e1(d(e(d(d(e1(e(x))))))))))
                                             e(d(d(e1(d(e(d(d(e1(d1(x))))))))))
RW143,1 & A16      --> * RW144:  All x:Any + =(d1(d(d(e1(d(e(d(d(x))))))))
                                             e(d(d(e1(d(e(d(d(e1(d1(x))))))))))
RW144,1 & A14      --> * RW145:  All x:Any + =(d(e1(d(e(d(d(x)))))) e(d(d(e1(d(e(d(d(e1(d1(x))))))))))
RW120,1 & RW119    --> * RW153:  All x:Any + =(e1(d1(e(x))) d(d(e1(e(d(d(e1(d(e(d(d(e1(d(x))))))))))))))
RW153,1 & A16      --> * RW154:  All x:Any + =(e1(d1(e(x))) d(d(d(d(e1(d(e(d(d(e1(d(x))))))))))))
P113,1 & RW119     --> * RW155:  All x:Any + =(e(d(d(e1(d(e(d(d(e1(d(e(x))))))))))) d(d(e1(x))))
A15,1 & RW154,1    --> * P158:   All x:Any + =(e1(d1(x)) d(d(d(d(e1(d(e(d(d(e1(d(e1(x))))))))))))))
RW145,1 & P158     --> * RW160:
  All x:Any + =(d(e1(d(e(d(d(x)))))) e(d(d(e1(d(e(d(d(d(d(d(d(e1(d(e(d(d(e1(d(e1(x))))))))))))))))))))
A15,1 & RW155,1    --> * P163:   All x:Any + =(e(d(d(e1(d(e(d(d(e1(d(x))))))))))) d(d(e1(e1(x)))))
RW124,1 & A14,1    --> * P167:   All x:Any + =(d1(d(x)) d(e1(d(e(d(d(d(d(e1(d(e(d(d(e1(x))))))))))))))))
P167,1 & A14       --> * RW168:  All x:Any + =(x d(e1(d(e(d(d(d(d(e1(d(e(d(d(e1(x))))))))))))))))
P158,1 & A15,1     --> * P170:   All x:Any + =(e(d(d(d(d(e1(d(e(d(d(e1(d(e1(x))))))))))))))) d1(x))
A13,1 & P170       --> * RW171:  All x:Any + =(d(e(d(d(d(d(e1(d(e(d(d(e1(d(e1(x))))))))))))))) x)
A14,1 & P170       --> * RW172:  All x:Any + =(e(d(d(d(d(e1(d(e(d(d(e1(d(e1(d(x))))))))))))))) x)
```

```
P163,1 & A16,1     --> * P175:   All x:Any + =(e1(d(d(e1(e1(x))))) d(d(e1(d(e(d(d(e1(d(x))))))))))
RW171,1 & P175     --> * RW176:  All x:Any + =(d(e(d(d(e1(d(d(e1(e1(e1(x))))))))))) x)
RW172,1 & P175     --> * RW177:  All x:Any + =(e(d(d(e1(d(d(e1(e1(e1(d(x))))))))))) x)
RW160,1 & P175     --> * RW179:  All x:Any + =(d(e1(d(e(d(d(x))))))
                                             e(d(d(e1(d(e(d(d(d(d(e1(d(d(e1(e1(e1(x))))))))))))))))))
RW177,1 & A16,1    --> * P184:   All x:Any + =(e1(x) d(d(e1(d(d(e1(e1(e1(d(x))))))))))
A16,1 & RW176,1    --> * P187:   All x:Any + =(d(e(d(d(e1(d(d(e1(e1(x))))))))) e(x))
P187,1 & P175,1    --> * P190:   All x:Any + =(e1(d(d(e1(e1(d(e1(e1(x))))))))) d(d(e1(e(x)))))
P190,1 & A16       --> * RW191:  All x:Any + =(e1(d(d(e1(e1(d(e1(e1(x))))))))) d(d(x)))
A16,1 & P187,1     --> * P192:   All x:Any + =(d(e(d(d(e1(d(d(e1(x))))))))) e(e(x)))
RW191,1 & A15,1    --> * P200:   All x:Any + =(e(d(d(x))) d(d(e1(e1(d(e1(e1(x))))))))
A16,1 & P192,1     --> * P205:   All x:Any + =(d(e(d(d(e1(d(d(x))))))) e(e(e(x))))
A16,1 & P200,1     --> * P208:   All x:Any + =(e(d(d(e(x)))) d(d(e1(e1(d(e1(x)))))))
P208,1 & P205,1    --> * P215:   All x:Any + =(d(e(d(d(e1(e(d(d(e(x))))))))) e(e(e(e1(e1(d(e1(x))))))))
P215,1 & A16       --> * RW216:  All x:Any + =(d(e(d(d(d(d(e(x))))))) e(e(e(e1(e1(d(e1(x))))))))
RW216,1 & A15      --> * RW217:  All x:Any + =(d(e(d(d(d(d(e(x))))))) e(e(e1(d(e1(x))))))
RW217,1 & A15      --> * RW218:  All x:Any + =(d(e(d(d(d(d(e(x))))))) e(d(e1(x))))
A16,1 & P208,1     --> * P219:   All x:Any + =(e(d(d(e(e(x))))) d(d(e1(e1(d(x))))))
A15,1 & RW218,1    --> * P224:   All x:Any + =(d(e(d(d(d(d(x))))))) e(d(e1(e1(x)))))
RW179,1 & P224     --> * RW225:  All x:Any + =(d(e1(d(e(d(d(x))))))
                                             e(d(d(e1(e(d(e1(e1(e1(d(d(e1(e1(e1(x))))))))))))))))
RW225,1 & A16      --> * RW226:  All x:Any + =(d(e1(d(e(d(d(x))))))
                                             e(d(d(d(e1(e1(e1(d(d(e1(e1(e1(x))))))))))))))
RW168,1 & P224     --> * RW227:  All x:Any + =(x d(e1(e(d(e1(e1(e1(d(e(d(d(e1(x))))))))))))))
RW227,1 & A16      --> * RW228:  All x:Any + =(x d(d(e1(e1(e1(d(e(d(d(e1(x))))))))))))
A16,1 & RW228,1    --> * P231:   All x:Any + =(e(x) d(d(e1(e1(e1(d(e(d(d(x))))))))))
P175,1 & P224,1    --> * P234:   All x:Any + =(d(e(d(d(e1(d(d(e1(e1(x))))))))))
                                             e(d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
P234,1 & P205      --> * RW235:  All x:Any + =(e(e(e(e1(e1(x))))) e(d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
RW235,1 & A15      --> * RW236:  All x:Any + =(e(e(e1(x))) e(d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
RW236,1 & A15      --> * RW237:  All x:Any + =(e(x) e(d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
RW237,1 & A16,1    --> * P238:   All x:Any + =(e1(e(x)) d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
P238,1 & A16       --> * RW239:  All x:Any + =(x d(e1(e1(e1(d(e(d(d(e1(d(x))))))))))))
P184,1 & P205,1    --> * P241:   All x:Any + =(d(e(d(d(e1(e1(x))))))) e(e(e(e1(d(d(e1(e1(e1(d(x))))))))))
P241,1 & A15       --> * RW242:  All x:Any + =(d(e(d(d(e1(e1(x))))))) e(e(d(d(e1(e1(e1(d(x))))))))
P205,1 & P219,1    --> * P247:   All x:Any + =(e(d(d(e(e(e(d(d(e1(d(d(x))))))))))))
                                             d(d(e1(e1(e(e(e(x))))))))
P247,1 & A16       --> * RW248:  All x:Any + =(e(d(d(e(e(e(d(d(e1(d(d(x))))))))))) d(d(e1(e(e(x))))))
RW248,1 & A16      --> * RW249:  All x:Any + =(e(d(d(e(e(e(d(d(e1(d(d(x))))))))))) d(d(e(x))))
P175,1 & RW239,1   --> * P252:   All x:Any + =(e(d(d(e1(d(x)))))
                                             d(e1(e1(e1(d(e(e1(d(d(e1(e1(x))))))))))))
P252,1 & A15       --> * RW253:  All x:Any + =(e(d(d(e1(d(x))))) d(e1(e1(e1(d(d(d(e1(e1(x))))))))))
A16,1 & RW242,1    --> * P254:   All x:Any + =(d(e(d(d(e1(x))))) e(e(d(d(e1(e1(e1(d(e(x))))))))))
P175,1 & P254      --> * RW269:  All x:Any + =(e1(d(d(e1(e1(x))))))
                                             d(d(e1(e(e(d(d(e1(e1(e1(d(e(d(x))))))))))))))
RW269,1 & A16      --> * RW270:  All x:Any + =(e1(d(d(e1(e1(x))))) d(d(e(d(d(e1(e1(e1(d(e(d(x))))))))))))
RW270,1 & P254     --> * RW271:  All x:Any + =(e1(d(d(e1(e1(x))))))
                                             d(e(e(d(d(e1(e1(e1(d(e(e1(e1(d(e(d(x))))))))))))))))
RW271,1 & A15      --> * RW272:  All x:Any + =(e1(d(d(e1(e1(x))))))
                                             d(e(e(d(d(e1(e1(e1(d(e1(d(e(d(x))))))))))))))
RW239,1 & P254     --> * RW275:  All x:Any + =(x d(e1(e1(e1(e(e(d(d(e1(e1(e1(d(e(d(x))))))))))))))))
RW275,1 & A16      --> * RW276:  All x:Any + =(x d(e1(e1(e(d(d(e1(e1(e1(d(e(d(x))))))))))))))
RW276,1 & A16      --> * RW277:  All x:Any + =(x d(e1(d(d(e1(e1(e1(d(e(d(x))))))))))))
RW249,1 & A16,1    --> * P283:   All x:Any + =(e1(d(d(e(x)))) d(d(e(e(e(d(d(e1(d(d(x))))))))))))
A16,1 & RW253,1    --> * P289:   All x:Any + =(e(d(d(e1(d(e(x))))))) d(e1(e1(e1(d(d(d(e1(x))))))))))
A16,1 & P254,1     --> * P292:   All x:Any + =(d(e(d(d(x)))) e(e(d(d(e1(e1(e1(d(e(e(x))))))))))))
A16,1 & P289,1     --> * P302:   All x:Any + =(e(d(d(e1(d(e(e(x))))))))) d(e1(e1(e1(d(d(d(x))))))))
P292,1 & A16,1     --> * P305:   All x:Any + =(e1(d(e(d(d(x))))))) e(d(d(e1(e1(e1(d(e(e(x))))))))))
P305,1 & A16,1     --> * P308:   All x:Any + =(e1(e1(d(e(d(d(x))))))) d(d(e1(e1(e1(d(e(e(x))))))))))
P224,1 & P219,1    --> * P314:   All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))))
                                             d(d(e1(e1(e(d(e1(e1(x))))))))))
P314,1 & A16       --> * RW315:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))) d(d(e1(d(e1(e1(x))))))))
P219,1 & P224,1    --> * P316:   All x:Any + =(d(e(d(d(e(d(d(e(e(x))))))))))) e(d(e1(e1(e1(e1(d(x))))))))
P231,1 & RW226,1   --> * P317:   All x:Any + =(d(e1(d(e(d(d(d(e(d(d(x))))))))))))
                                             e(d(d(d(e1(e1(e1(e(x))))))))))
```

```
P317,1 & A16       --> * RW318:  All x:Any + =(d(e1(d(e(d(d(d(e(d(d(x))))))))) e(d(d(d(e1(e1(x)))))))
P231,1 & P184,1    --> * P319:   All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))
                                              d(d(e1(d(d(e1(e1(e1(e(x))))))))))
P319,1 & A16       --> * RW320:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))
                                              d(d(e1(d(d(e1(e1(x)))))))))
P231,1 & P219,1    --> * P331:   All x:Any + =(e(d(d(e(e(d(e1(e1(e1(d(e(d(d(x))))))))))))))
                                              d(d(e1(e1(e(x))))))
P331,1 & A16       --> * RW332:  All x:Any + =(e(d(d(e(e(d(e1(e1(e1(d(e(d(d(x))))))))))))))) d(d(e1(x))))
P224,1 & RW277,1   --> * P334:   All x:Any + =(e(d(d(d(d(x)))))
                                              d(e1(d(d(e1(e1(e1(d(e(e(d(e1(e1(x))))))))))))))
P334,1 & P308      --> * RW335:  All x:Any + =(e(d(d(d(d(x))))) d(e1(e1(e1(d(e(d(d(d(e1(e1(x))))))))))))
P231,1 & P254,1    --> * P336:   All x:Any + =(d(e(e(x)))
                                              e(e(d(d(e1(e1(e1(d(e(e1(e1(d(e(d(d(x))))))))))))))))
P336,1 & A15       --> * RW337:  All x:Any + =(d(e(e(x))) e(e(d(d(e1(e1(e1(d(e1(d(e(d(d(x))))))))))))))
P219,1 & P254,1    --> * P338:   All x:Any + =(d(e(e(d(d(e(e(x)))))))
                                              e(e(d(d(e1(e1(e1(d(e(e1(d(x))))))))))))
P338,1 & A15       --> * RW339:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) e(e(d(d(e1(e1(e1(d(d(x))))))))))
P302,1 & RW339,1   --> * P340:   All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(d(d(e1(e1(e1(d(e(d(d(e1(d(e(e(x))))))))))))))))
P340,1 & P254      --> * RW341:
  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                e(e(d(d(e1(e1(e1(e(e(d(d(e1(e1(e1(d(e(d(e(e(x))))))))))))))))))))
RW341,1 & A16      --> * RW342:
  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                e(e(d(d(e1(e1(e(d(d(e1(e1(e1(d(e(d(e(e(x))))))))))))))))))
RW342,1 & A16      --> * RW343:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(d(d(e1(d(d(e1(e1(e1(d(e(d(e(e(x))))))))))))))))
RW343,1 & RW320    --> * RW344:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(e1(d(e1(e1(e1(d(e(d(d(e1(d(e(d(e(e(x))))))))))))))))))
RW344,1 & P254     --> * RW345:
  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                e(e(e1(d(e1(e1(e1(e(e(d(d(e1(e1(e1(d(e(d(e(d(e(e(x))))))))))))))))))))))
RW345,1 & A16      --> * RW346:
  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                e(e(e1(d(e1(e1(e(d(d(e1(e1(e1(d(e(d(e(d(e(e(x))))))))))))))))))))
RW346,1 & A16      --> * RW347:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(e1(d(e1(d(d(e1(e1(e1(d(e(d(e(d(e(e(x))))))))))))))))))
RW347,1 & RW277    --> * RW348:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(e1(e(d(e(e(x)))))))
RW348,1 & A16      --> * RW349:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(d(d(x))))))))))))))
                                              e(e(d(e(e(x))))))
RW349,1 & A15      --> * RW350:  All x:Any + =(d(e(e(d(d(e(e1(e1(d(d(d(x))))))))))) e(e(d(e(e(x))))))
RW350,1 & A15      --> * RW351:  All x:Any + =(d(e(e(d(d(e1(d(d(d(x))))))))) e(e(d(e(e(x))))))
P302,1 & RW277,1   --> * P359:   All x:Any + =(e1(e1(e1(d(d(d(x))))))
                                              d(e1(d(d(e1(e1(e1(d(e(e(d(d(e1(d(e(e(x))))))))))))))))
P359,1 & P308      --> * RW360:  All x:Any + =(e1(e1(e1(d(d(d(x))))))
                                              d(e1(e1(e1(d(e(d(d(d(d(e1(d(e(e(x))))))))))))))))
RW360,1 & P224     --> * RW361:  All x:Any + =(e1(e1(e1(d(d(d(x))))))
                                              d(e1(e1(e1(e(d(e1(e1(e1(d(e(e(x))))))))))))))
RW361,1 & A16      --> * RW362:  All x:Any + =(e1(e1(e1(d(d(d(x))))))
                                              d(e1(e1(d(e1(e1(e1(d(e(e(x))))))))))))
P231,1 & P302,1    --> * P363:   All x:Any + =(e(d(d(e1(d(e(e(e1(e1(e1(d(e(d(d(x))))))))))))))
                                              d(e1(e1(e1(d(e(x)))))))
P363,1 & A15       --> * RW364:  All x:Any + =(e(d(d(e1(d(e(e1(e1(d(e(d(d(x))))))))))))
                                              d(e1(e1(e1(d(e(x)))))))
RW364,1 & A15      --> * RW365:  All x:Any + =(e(d(d(e1(d(e1(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
P308,1 & P224,1    --> * P369:   All x:Any + =(d(e(d(d(d(e1(e1(d(e(d(d(x))))))))))))
                                              e(d(e1(e1(d(e1(e1(e1(d(e(e(x))))))))))))
P369,1 & RW362     --> * RW370:  All x:Any + =(d(e(d(d(d(e1(e1(d(e(d(d(x))))))))))))
                                              e(e1(e1(e1(d(d(d(x)))))))
RW370,1 & P219     --> * RW371:  All x:Any + =(d(e(d(e(d(d(d(e(e(e(d(d(x))))))))))))
                                              e(e1(e1(e1(d(d(d(x)))))))
RW371,1 & A15      --> * RW372:  All x:Any + =(d(e(d(e(d(d(d(e(e(e(d(d(x))))))))))))) e1(e1(d(d(d(x))))))
P283,1 & RW372,1   --> * P373:   All x:Any + =(d(e(d(e(e1(d(d(e(x)))))))) e1(e1(d(d(d(e1(d(d(x))))))))
P373,1 & A15       --> * RW374:  All x:Any + =(d(e(d(d(d(e(x)))))) e1(e1(d(d(d(e1(d(d(x))))))))
```

```
P231,1 & RW372,1   --> * P375:   All x:Any + =(d(e(d(e(d(d(e(e(e(e(x)))))))))
                                               e1(e1(d(d(d(e1(e1(e1(d(e(d(d(x))))))))))))))
P375,1 & P231      --> * RW376:  All x:Any + =(d(e(d(e(d(d(e(e(e(e(x)))))))))) e1(e1(d(e(x)))))
A15,1 & RW376,1    --> * P377:   All x:Any + =(d(e(d(e(d(d(e(e(e(e(x)))))))))) e1(e1(d(e(e1(x))))))
P377,1 & A15       --> * RW378:  All x:Any + =(d(e(d(e(d(d(e(e(e(e(x)))))))))) e1(e1(d(x))))
A15,1 & RW378,1    --> * P379:   All x:Any + =(d(e(d(e(d(d(e(e(e(x)))))))) e1(e1(d(e1(x)))))
A15,1 & P379,1     --> * P382:   All x:Any + =(d(e(d(e(d(d(e(e(x)))))))) e1(e1(d(e1(e1(x))))))
A15,1 & P382,1     --> * P385:   All x:Any + =(d(e(d(e(d(d(x)))))) e1(e1(d(e1(e1(e1(x)))))))
P219,1 & P231,1    --> * P388:   All x:Any + =(e(d(e1(e1(d(x)))))
                                               d(d(e1(e1(e1(d(e(d(e(d(d(e(e(x))))))))))))))
P388,1 & P385      --> * RW389:  All x:Any + =(e(d(e1(e1(d(x)))))
                                               d(d(e1(e1(e1(e1(e1(d(e1(e1(e1(e(e(x))))))))))))))
RW389,1 & A16      --> * RW390:  All x:Any + =(e(d(e1(e1(d(x)))))
                                               d(d(e1(e1(e1(e1(e1(d(e1(e1(e(x))))))))))))
RW390,1 & A16      --> * RW391:  All x:Any + =(e(d(e1(e1(d(x))))) d(d(e1(e1(e1(e1(e1(d(e1(x))))))))))
P219,1 & RW335,1   --> * P392:   All x:Any + =(e(d(d(d(d(d(x))))))
                                               d(e1(e1(e1(d(e(d(e(d(d(e(e(x))))))))))))))
P392,1 & P385      --> * RW393:  All x:Any + =(e(d(d(d(d(d(x))))))
                                               d(e1(e1(e1(e1(e1(d(e1(e1(e1(e(e(x))))))))))))))
RW393,1 & A16      --> * RW394:  All x:Any + =(e(d(d(d(d(d(x))))))
                                               d(e1(e1(e1(e1(e1(d(e1(e1(e(x))))))))))))
RW394,1 & A16      --> * RW395:  All x:Any + =(e(d(d(d(d(d(x))))))) d(e1(e1(e1(e1(e1(d(e1(x))))))))))
RW374,1 & A15,1    --> * P396:   All x:Any + =(e(d(e(d(d(d(e(x)))))))) e1(d(d(d(e1(d(d(x))))))))
P385,1 & P224,1    --> * P399:   All x:Any + =(d(e(d(d(d(d(e1(e1(d(e1(e1(e1(x))))))))))))
                                               e(d(e1(e1(e(d(e(d(d(d(x))))))))))
P399,1 & P219      --> * RW400:  All x:Any + =(d(e(d(e(d(d(e(e(e1(e1(e1(x))))))))))))
                                               e(d(e1(e1(e(d(e(d(d(d(x))))))))))
RW400,1 & P385     --> * RW401:  All x:Any + =(e1(e1(d(e1(e1(e1(e(e(e1(e1(e1(x))))))))))))
                                               e(d(e1(e1(e(d(e(d(d(d(x))))))))))
RW401,1 & A16      --> * RW402:  All x:Any + =(e1(e1(d(e1(e1(e(e1(e1(e1(x))))))))))
                                               e(d(e1(e1(e(d(e(d(d(d(x))))))))))
RW402,1 & A16      --> * RW403:  All x:Any + =(e1(e1(d(e1(e1(e1(e1(x)))))))
                                               e(d(e1(e1(e(d(e(d(d(d(x))))))))))
RW403,1 & A16      --> * RW404:  All x:Any + =(e1(e1(d(e1(e1(e1(e1(x)))))))) e(d(e1(d(e(d(d(x)))))))
P231,1 & RW351,1   --> * P405:   All x:Any + =(d(e(e(d(d(e1(d(e(x)))))))
                                               e(e(d(e(e(e1(e1(e1(d(e(d(d(x))))))))))))))
P405,1 & A15       --> * RW406:  All x:Any + =(d(e(e(d(d(e1(d(e(x)))))))
                                               e(e(d(e(e1(e1(d(e(d(d(x))))))))))))
RW406,1 & A15      --> * RW407:  All x:Any + =(d(e(e(d(d(e1(d(e(x)))))))) e(e(d(e1(d(e(d(d(x))))))))
RW407,1 & RW404    --> * RW408:  All x:Any + =(d(e(e(d(d(e1(d(e(x)))))))
                                               e(e1(e1(d(e1(e1(e1(e1(x))))))))
RW408,1 & A15      --> * RW409:  All x:Any + =(d(e(e(d(d(e1(d(e(x)))))))) e1(d(e1(e1(e1(e1(x)))))))
P231,1 & RW339,1   --> * P410:   All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(e(d(d(x))))))))))))))
                                               e(e(d(d(e1(e1(e1(e(x))))))))
P410,1 & A16       --> * RW411:  All x:Any + =(d(e(e(d(d(e(e(e1(e1(e1(d(e(d(d(x))))))))))))))
                                               e(e(d(d(e1(e1(x)))))))
RW411,1 & A15      --> * RW412:  All x:Any + =(d(e(e(d(d(e(e1(e1(d(e(d(d(x))))))))))))
                                               e(e(d(d(e1(e1(x)))))))
RW412,1 & A15      --> * RW413:  All x:Any + =(d(e(e(d(d(e1(d(e(d(d(x))))))))))) e(e(d(d(e1(e1(x)))))))
RW413,1 & RW409    --> * RW414:  All x:Any + =(e1(d(e1(e1(e1(e1(d(d(x))))))))) e(e(d(d(e1(e1(x)))))))
RW320,1 & P385,1   --> * P415:   All x:Any + =(d(e(d(e(e1(d(e1(e1(e1(d(e(d(d(x))))))))))))))
                                               e1(e1(d(e1(e1(e1(e1(d(d(e1(e1(x))))))))))))
P415,1 & RW414     --> * RW416:  All x:Any + =(d(e(d(e(e1(d(e1(e1(e1(d(e(d(d(x))))))))))))))
                                               e1(e(e(d(d(e1(e1(e1(e1(x))))))))))
RW416,1 & A16      --> * RW417:  All x:Any + =(d(e(d(e(e1(d(e1(e1(e1(d(e(d(d(x))))))))))))))
                                               e(d(d(e1(e1(e1(e1(x)))))))
RW417,1 & A15      --> * RW418:  All x:Any + =(d(e(d(d(e1(e1(e1(d(e(d(d(x))))))))))))
                                               e(d(d(e1(e1(e1(e1(x)))))))
RW418,1 & P254     --> * RW419:  All x:Any + =(e(e(d(d(e1(e1(e1(d(e(e1(e1(d(e(d(d(x))))))))))))))))
                                               e(d(d(e1(e1(e1(e1(x)))))))
RW419,1 & A15      --> * RW420:  All x:Any + =(e(e(d(d(e1(e1(e1(d(e1(d(e(d(d(x))))))))))))))
                                               e(d(d(e1(e1(e1(e1(x)))))))
RW420,1 & RW337    --> * RW421:  All x:Any + =(d(e(e(x))) e(d(d(e1(e1(e1(e1(x)))))))
RW421,1 & A16,1    --> * P422:   All x:Any + =(e1(d(e(e(x)))) d(d(e1(e1(e1(e1(x)))))))
RW391,1 & P422     --> * RW423:  All x:Any + =(e(d(e1(e1(d(x))))) e1(d(e(e(e1(d(e1(x))))))))
```

```
RW423,1 & A15      --> * RW424:  All x:Any + =(e(d(e1(e1(d(x))))) e1(d(e(d(e1(x))))))
P422,1 & RW318,1   --> * P427:   All x:Any + =(d(e1(d(e(d(d(d(e(e1(d(e(e(x))))))))))))
                                              e(d(d(d(e1(e1(e1(e1(e1(e1(x))))))))))
P427,1 & P422      --> * RW428:  All x:Any + =(d(e1(d(e(d(d(d(e(e1(d(e(e(x))))))))))))
                                              e(d(e1(d(e(e(e1(e1(x))))))))
RW428,1 & A15      --> * RW429:  All x:Any + =(d(e1(d(e(d(d(d(d(e(e(x))))))))))
                                              e(d(e1(d(e(e(e1(e1(x))))))))
RW429,1 & P224     --> * RW430:  All x:Any + =(d(e1(e(d(e1(e1(e(e(x))))))))) e(d(e1(d(e(e(e1(e1(x))))))))
RW430,1 & A16      --> * RW431:  All x:Any + =(d(d(e1(e1(e(e(x)))))) e(d(e1(d(e(e(e1(e1(x))))))))
RW431,1 & A16      --> * RW432:  All x:Any + =(d(d(e1(e(x)))) e(d(e1(d(e(e(e1(e1(x))))))))
RW432,1 & A16      --> * RW433:  All x:Any + =(d(d(x)) e(d(e1(d(e(e(e1(e1(x))))))))
RW433,1 & A15      --> * RW434:  All x:Any + =(d(d(x)) e(d(e1(d(e(e1(x)))))))
RW434,1 & A15      --> * RW435:  All x:Any + =(d(d(x)) e(d(e1(d(x)))))
RW404,1 & RW435    --> * RW436:  All x:Any + =(e1(e1(d(e1(e1(e1(e1(x)))))))) d(d(e(d(d(x))))))
P316,1 & RW436     --> * RW437:  All x:Any + =(d(e(e1(e1(d(e1(e1(e1(e1(e(e(x))))))))))))
                                              e(d(e1(e1(e1(e1(d(x)))))))
RW437,1 & A16      --> * RW438:  All x:Any + =(d(e(e1(e1(d(e1(e1(e1(e(x))))))))))
                                              e(d(e1(e1(e1(e1(d(x)))))))
RW438,1 & A16      --> * RW439:  All x:Any + =(d(e(e1(e1(d(e1(e1(x)))))))) e(d(e1(e1(e1(e1(d(x)))))))
RW439,1 & A15      --> * RW440:  All x:Any + =(d(e1(d(e1(e1(x))))) e(d(e1(e1(e1(e1(d(x)))))))
RW315,1 & RW440    --> * RW443:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))
                                              d(e(d(e1(e1(e1(e1(d(x))))))))
RW277,1 & RW435,1  --> * P447:   All x:Any + =(d(d(e1(d(d(e1(e1(e1(d(e(d(x))))))))))) e(d(e1(x))))
P447,1 & RW320     --> * RW448:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(e1(d(e(d(x))))))))))))) e(d(e1(x))))
RW448,1 & P254     --> * RW449:
  All x:Any + =(e1(d(e1(e1(e1(e(e(d(d(e1(e1(e1(d(e(d(e(d(x))))))))))))))))) e(d(e1(x))))
RW449,1 & A16      --> * RW450:  All x:Any + =(e1(d(e1(e1(e(d(d(e1(e1(e1(d(e(d(e(d(x))))))))))))))))
                                              e(d(e1(x))))
RW450,1 & A16      --> * RW451:  All x:Any + =(e1(d(e1(d(d(e1(e1(e1(d(e(d(e(d(x))))))))))))))) e(d(e1(x))))
RW451,1 & RW277    --> * RW452:  All x:Any + =(e1(e(d(x))) e(d(e1(x))))
RW452,1 & A16      --> * RW453:  All x:Any + =(d(x) e(d(e1(x))))
P224,1 & RW453     --> * RW454:  All x:Any + =(d(e(d(d(d(d(x)))))) d(e1(x)))
RW443,1 & RW453    --> * RW455:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))) d(d(e1(e1(e1(d(x)))))))
RW440,1 & RW453    --> * RW456:  All x:Any + =(d(e1(d(e1(e1(x))))) d(e1(e1(e1(d(x))))))
RW332,1 & RW453    --> * RW462:  All x:Any + =(e(d(d(e(d(e1(e1(d(e(d(d(x))))))))))) d(d(e1(x))))
RW462,1 & RW453    --> * RW463:  All x:Any + =(e(d(d(d(e1(d(e(d(d(d(x))))))))))) d(d(e1(x))))
RW424,1 & RW453    --> * RW465:  All x:Any + =(d(e1(d(x))) e1(d(e(d(e1(x))))))
RW465,1 & RW453    --> * RW466:  All x:Any + =(d(e1(d(x))) e1(d(d(x))))
RW126,1 & RW466    --> * RW470:  - =(d(c_1) e(d(e1(d(d(e(d(d(e(c_1)))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW470,1 & RW466    --> * RW471:  - =(d(c_1) e(e1(d(d(d(e(d(d(e(c_1)))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW471,1 & RW436    --> * RW472:  - =(d(c_1) e(e1(d(e1(e1(d(e1(e1(e1(e1(e(c_1))))))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW472,1 & A16      --> * RW473:  - =(d(c_1) e(e1(d(e1(e1(d(e1(e1(e1(c_1)))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW473,1 & A15      --> * RW474:  - =(d(c_1) d(e1(e1(d(e1(e1(e1(c_1))))))))
                                 - =(d(c_2) e(e(e(e(e(c_2))))))
RW272,1 & RW466    --> * RW488:  All x:Any + =(e1(d(d(e1(e1(x))))))
                                              d(e(e(d(d(e1(e1(e1(e1(d(d(e(d(x))))))))))))))
RW488,1 & P422     --> * RW489:  All x:Any + =(e1(d(d(e1(e1(x))))) d(e(e(e1(d(e(e(d(d(e(d(x))))))))))))
RW489,1 & A15      --> * RW490:  All x:Any + =(e1(d(d(e1(e1(x))))) d(e(d(e(e(d(d(e(d(x)))))))))
RW277,1 & RW466    --> * RW491:  All x:Any + =(x e1(d(d(d(e1(e1(e1(d(e(d(x))))))))))))
RW456,1 & RW466    --> * RW498:  All x:Any + =(e1(d(d(e1(e1(x))))) d(e1(e1(e1(d(x))))))
RW320,1 & RW466    --> * RW499:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              d(e1(d(d(d(e1(e1(x))))))))
RW499,1 & RW466    --> * RW500:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(d(d(d(e1(e1(x))))))))
RW463,1 & RW466    --> * RW501:  All x:Any + =(e(d(d(e1(d(d(e(d(d(x))))))))) d(d(e1(x))))
RW501,1 & RW466    --> * RW502:  All x:Any + =(e(d(e1(d(d(d(e(d(d(x))))))))) d(d(e1(x))))
RW502,1 & RW466    --> * RW503:  All x:Any + =(e(e1(d(d(d(d(e(d(d(x))))))))) d(d(e1(x))))
RW503,1 & RW436    --> * RW504:  All x:Any + =(e(e1(d(d(e1(e1(e1(d(e1(e1(e1(e1(x))))))))))))) d(d(e1(x))))
RW504,1 & P219     --> * RW505:  All x:Any + =(e(e1(e(d(d(e(e(e1(e1(e1(e1(x))))))))))))) d(d(e1(x))))
RW505,1 & A16      --> * RW506:  All x:Any + =(e(d(d(e(e(e1(e1(e1(e1(x))))))))))) d(d(e1(x))))
RW506,1 & A15      --> * RW507:  All x:Any + =(e(d(d(e(e1(e1(e1(x)))))))) d(d(e1(x))))
```

```
RW507,1 & A15      --> * RW508:  All x:Any + =(e(d(d(e1(e1(x))))) d(d(e1(x))))
RW351,1 & RW466    --> * RW517:  All x:Any + =(d(e(e(d(e1(d(d(d(d(x))))))))) e(e(d(e(e(x))))))
RW517,1 & RW466    --> * RW518:  All x:Any + =(d(e(e(e1(d(d(d(d(d(x))))))))) e(e(d(e(e(x))))))
RW518,1 & A15      --> * RW519:  All x:Any + =(d(e(d(d(d(d(d(x)))))))) e(e(d(e(e(x))))))
RW519,1 & RW454    --> * RW520:  All x:Any + =(d(e1(d(x))) e(e(d(e(e(x))))))
RW520,1 & RW466    --> * RW521:  All x:Any + =(e1(d(d(x))) e(e(d(e(e(x))))))
RW365,1 & RW466    --> * RW533:  All x:Any + =(e(d(e1(d(d(e1(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
RW533,1 & RW466    --> * RW534:  All x:Any + =(e(e1(d(d(d(e1(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
RW534,1 & RW466    --> * RW535:  All x:Any + =(e(e1(d(d(e1(d(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
RW535,1 & RW466    --> * RW536:  All x:Any + =(e(e1(d(e1(d(d(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
RW536,1 & RW466    --> * RW537:  All x:Any + =(e(e1(e1(d(d(d(d(e(d(d(x))))))))))) d(e1(e1(e1(d(e(x)))))))
RW537,1 & RW436    --> * RW538:  All x:Any + =(e(e1(e1(d(d(e1(e1(d(e1(e1(e1(e1(x))))))))))))
                                              d(e1(e1(e1(d(e(x)))))))
RW538,1 & P219     --> * RW539:  All x:Any + =(e(e1(e1(e(d(d(e(e(e1(e1(e1(e1(x))))))))))))
                                              d(e1(e1(e1(d(e(x)))))))
RW539,1 & A16      --> * RW540:  All x:Any + =(e(e1(d(d(e(e(e1(e1(e1(e1(x))))))))))
                                              d(e1(e1(e1(d(e(x)))))))
RW540,1 & A15      --> * RW541:  All x:Any + =(d(d(e(e(e1(e1(e1(e1(x)))))))) d(e1(e1(e1(d(e(x)))))))
RW541,1 & A15      --> * RW542:  All x:Any + =(d(d(e(e1(e1(e1(x))))))) d(e1(e1(e1(d(e(x)))))))
RW542,1 & A15      --> * RW543:  All x:Any + =(d(d(e1(e1(e1(x))))) d(e1(e1(e1(d(e(x)))))))
P396,1 & RW466     --> * RW547:  All x:Any + =(e(d(e(d(d(d(e(x)))))))) e1(d(d(e1(d(d(d(x)))))))))
RW547,1 & RW466    --> * RW548:  All x:Any + =(e(d(e(d(d(d(e(x)))))))) e1(d(e1(d(d(d(d(x)))))))))
RW548,1 & RW466    --> * RW549:  All x:Any + =(e(d(e(d(d(d(e(x)))))))) e1(e1(d(d(d(d(d(x)))))))))
RW409,1 & RW466    --> * RW550:  All x:Any + =(d(e(e(d(e1(d(d(e(x)))))))) e1(d(e1(e1(e1(e1(x)))))))
RW550,1 & RW466    --> * RW551:  All x:Any + =(d(e(e(e1(d(d(d(e(x)))))))) e1(d(e1(e1(e1(e1(x)))))))
RW551,1 & A15      --> * RW552:  All x:Any + =(d(e(d(d(d(e(x)))))) e1(d(e1(e1(e1(e1(x)))))))
RW549,1 & RW552    --> * RW553:  All x:Any + =(e(e1(d(e1(e1(e1(e1(x)))))))) e1(e1(d(d(d(d(d(x)))))))))
RW553,1 & RW521    --> * RW554:  All x:Any + =(e(e1(d(e1(e1(e1(e1(x)))))))) e1(e(e(d(e(e(d(d(d(x))))))))))
RW554,1 & A16      --> * RW555:  All x:Any + =(e(e1(d(e1(e1(e1(e1(x)))))))) e(d(e(e(d(d(d(x)))))))))
RW555,1 & A15      --> * RW556:  All x:Any + =(d(e1(e1(e1(e1(x))))) e(d(e(e(d(d(d(x)))))))))
RW490,1 & RW543    --> * RW591:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))) d(e(d(e(e(d(d(e(d(x))))))))))
P219,1 & RW543     --> * RW592:  All x:Any + =(e(d(d(e(e(x))))) d(e1(e1(e1(d(e(d(x)))))))))
RW491,1 & RW543    --> * RW598:  All x:Any + =(x e1(d(d(e1(e1(e1(d(e(e1(d(e(d(x))))))))))))))
RW598,1 & RW543    --> * RW599:  All x:Any + =(x e1(d(e1(e1(e1(d(e(e1(d(e(e1(d(e(d(x))))))))))))))))
RW599,1 & A15      --> * RW600:  All x:Any + =(x e1(d(e1(e1(e1(d(d(e(e1(d(e(d(x))))))))))))))
RW600,1 & RW521    --> * RW601:  All x:Any + =(x e1(d(e1(e1(e(e(d(e(e(e(e1(d(e(d(x))))))))))))))))
RW601,1 & A16      --> * RW602:  All x:Any + =(x e1(d(e1(e(d(e(e(e(e(e1(d(e(d(x))))))))))))))
RW602,1 & A16      --> * RW603:  All x:Any + =(x e1(d(d(e(e(e(e1(d(e(d(x))))))))))))
RW603,1 & RW521    --> * RW604:  All x:Any + =(x e(e(d(e(e(e(e(e(e1(d(e(d(x))))))))))))))
RW604,1 & A15      --> * RW605:  All x:Any + =(x e(e(d(e(e(e(e(d(e(d(x))))))))))))
P308,1 & RW543     --> * RW612:  All x:Any + =(e1(e1(d(e(d(d(x)))))) d(e1(e1(e1(d(e(e1(d(e(e(x))))))))))))
RW612,1 & A15      --> * RW613:  All x:Any + =(e1(e1(d(e(d(d(x)))))) d(e1(e1(e1(d(d(e(e(x))))))))))
RW613,1 & RW521    --> * RW614:  All x:Any + =(e1(e1(d(e(d(d(x)))))) d(e1(e1(e(e(d(e(e(e(e(x))))))))))))
RW614,1 & A16      --> * RW615:  All x:Any + =(e1(e1(d(e(d(d(x)))))) d(e1(e(d(e(e(e(e(x))))))))))
RW615,1 & A16      --> * RW616:  All x:Any + =(e1(e1(d(e(d(d(x)))))) d(d(e(e(e(e(x)))))))
RW455,1 & RW543    --> * RW617:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))
                                              d(e1(e1(e1(d(e(e1(d(x))))))))))
RW617,1 & A15      --> * RW618:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x)))))))))) d(e1(e1(e1(d(d(x)))))))
RW618,1 & RW521    --> * RW619:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x))))))))))
                                              d(e1(e1(e(e(d(e(e(x))))))))))
RW619,1 & A16      --> * RW620:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x)))))))))) d(e1(e(d(e(e(x)))))))
RW620,1 & A16      --> * RW621:  All x:Any + =(e(d(d(e(e(e(d(d(d(d(x)))))))))) d(d(e(e(x)))))
RW498,1 & RW543    --> * RW622:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))) d(e1(e1(e1(d(x))))))
RW500,1 & RW543    --> * RW623:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(d(d(e1(e1(e1(d(e(x))))))))))
RW623,1 & RW543    --> * RW624:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(d(e1(e1(e1(d(e(e1(d(e(x))))))))))))
RW624,1 & RW543    --> * RW625:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(e1(e1(e1(d(e(e1(d(e(e1(d(e(x))))))))))))))
RW625,1 & A15      --> * RW626:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(e1(e1(e1(d(d(e(e1(d(e(x))))))))))))
RW626,1 & RW521    --> * RW627:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(e1(e1(e(e(d(e(e(e(e1(d(e(x))))))))))))))
RW627,1 & A16      --> * RW628:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x))))))))))
                                              e1(d(e1(e(d(e(e(e(e1(d(e(x))))))))))))
```

```
RW628,1 & A16      --> * RW629:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x)))))))))
                                               e1(d(d(e(e(e(e1(d(e(x)))))))))
RW629,1 & RW521    --> * RW630:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x)))))))))
                                               e(e(d(e(e(e(e(e(e1(d(e(x))))))))))))
RW630,1 & A15      --> * RW631:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x)))))))))
                                               e(e(d(e(e(e(e(d(e(x)))))))))
RW508,1 & RW543    --> * RW632:  All x:Any + =(e(d(e1(e1(e1(d(e(x)))))))) d(d(e1(x))))
RW632,1 & RW453    --> * RW633:  All x:Any + =(d(e1(e1(d(e(x))))) d(d(e1(x))))
RW335,1 & RW543    --> * RW634:  All x:Any + =(e(d(d(d(d(x)))))
                                               d(e1(e1(e1(d(e(d(d(d(e1(e1(e1(d(e(x))))))))))))))
RW634,1 & RW543    --> * RW635:  All x:Any + =(e(d(d(d(d(x)))))
                                               d(e1(e1(e1(d(e(d(e1(e1(e1(d(e(e1(d(e(x))))))))))))))))
RW635,1 & RW453    --> * RW636:  All x:Any + =(e(d(d(d(d(x)))))
                                               d(e1(e1(e1(d(d(e1(e1(d(e(e1(d(e(x))))))))))))))
RW636,1 & RW543    --> * RW637:  All x:Any + =(e(d(d(d(d(x)))))
                                               d(e1(e1(e1(d(e1(e1(e1(d(e(d(e(e1(d(e(x))))))))))))))))
RW637,1 & A15      --> * RW638:  All x:Any + =(e(d(d(d(d(x)))))
                                               d(e1(e1(e1(d(e1(e1(e1(d(e(d(d(e(x))))))))))))))
RW339,1 & RW543    --> * RW639:  All x:Any + =(d(e(e(d(d(e(e(x)))))))
                                               e(e(d(e1(e1(e1(d(e(e1(d(d(x))))))))))))
RW639,1 & RW521    --> * RW640:  All x:Any + =(d(e(e(d(d(e(e(x)))))))
                                               e(e(d(e1(e1(e1(d(e(e(e(d(e(e(x))))))))))))))
RW640,1 & RW453    --> * RW641:  All x:Any + =(d(e(e(d(d(e(e(x)))))))
                                               e(d(e1(e1(d(e(e(e(d(e(e(x))))))))))))
RW641,1 & RW453    --> * RW642:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) d(e1(d(e(e(e(d(e(e(x))))))))))
RW642,1 & RW466    --> * RW643:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) e1(d(d(e(e(e(d(e(e(x))))))))))
RW643,1 & RW521    --> * RW644:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) e(e(d(e(e(e(e(e(d(e(e(x))))))))))))
RW414,1 & RW543    --> * RW645:  All x:Any + =(e1(d(e1(e1(e1(e1(d(d(x)))))))))
                                               e(e(d(e1(e1(e1(d(e(x))))))))))
RW645,1 & RW521    --> * RW646:  All x:Any + =(e1(d(e1(e1(e1(e(e(d(e(e(x))))))))))
                                               e(e(d(e1(e1(e1(d(e(x))))))))))
RW646,1 & RW453    --> * RW647:  All x:Any + =(e1(d(e1(e1(e1(e(e(d(e(e(x))))))))))) e(d(e1(e1(d(e(x)))))))
RW647,1 & RW453    --> * RW648:  All x:Any + =(e1(d(e1(e1(e1(e(e(d(e(e(x))))))))))) d(e1(d(e(x)))))
RW648,1 & RW466    --> * RW649:  All x:Any + =(e1(d(e1(e1(e1(e(e(d(e(e(x))))))))))) e1(d(d(e(x)))))
RW649,1 & RW521    --> * RW650:  All x:Any + =(e1(d(e1(e1(e1(e(e(d(e(e(x))))))))))) e(e(d(e(e(e(x)))))))
RW650,1 & A16      --> * RW651:  All x:Any + =(e1(d(e1(e1(e(d(e(e(x))))))))) e(e(d(e(e(e(x)))))))
RW651,1 & A16      --> * RW652:  All x:Any + =(e1(d(e1(d(e(e(x)))))))) e(e(d(e(e(e(x)))))))
RW652,1 & RW466    --> * RW653:  All x:Any + =(e1(e1(d(d(e(e(x)))))))) e(e(d(e(e(e(x)))))))
RW653,1 & RW521    --> * RW654:  All x:Any + =(e1(e(e(d(e(e(e(e(x))))))))) e(e(d(e(e(e(x)))))))
RW654,1 & A16      --> * RW655:  All x:Any + =(e(d(e(e(e(e(x)))))))) e(e(d(e(e(e(x)))))))
P422,1 & RW543     --> * RW656:  All x:Any + =(e1(d(e(e(x)))))) d(e1(e1(e1(d(e(e1(e1(x))))))))))
RW656,1 & A15      --> * RW657:  All x:Any + =(e1(d(e(e(x)))))) d(e1(e1(e1(d(e1(x)))))))
RW638,1 & RW657    --> * RW658:  All x:Any + =(e(d(d(d(d(x))))) e1(d(e(e(e1(e1(d(e(d(d(e(x))))))))))))
RW658,1 & RW616    --> * RW659:  All x:Any + =(e(d(d(d(d(x))))) e1(d(e(e(d(d(e(e(e(e(e(x))))))))))))
RW659,1 & RW644    --> * RW660:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(d(e(e(e(e(e(d(e(e(e(e(e(x))))))))))))))))
RW660,1 & RW655    --> * RW661:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(d(e(e(e(e(e(d(e(e(e(e(e(x))))))))))))))))
RW661,1 & RW655    --> * RW662:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(e(d(e(e(e(e(d(e(e(e(e(e(x))))))))))))))))
RW662,1 & RW655    --> * RW663:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(e(d(e(e(e(e(d(e(e(e(e(e(x))))))))))))))))
RW663,1 & RW655    --> * RW664:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(e(e(d(e(e(e(d(e(e(e(e(e(x))))))))))))))))
RW664,1 & RW655    --> * RW665:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(e(e(d(e(e(e(e(e(d(e(e(e(e(x))))))))))))))))
RW665,1 & RW655    --> * RW666:  All x:Any + =(e(d(d(d(d(x)))))
                                               e1(e(e(e(e(e(e(d(e(e(e(d(e(e(e(e(x))))))))))))))))
RW666,1 & A16      --> * RW667:  All x:Any + =(e(d(d(d(d(x))))) e(e(e(e(e(d(e(e(e(e(d(e(e(e(e(x))))))))))))))))
RW605,1 & RW655    --> * RW669:  All x:Any + =(x e(e(e(d(e(e(e(e(d(e(d(x))))))))))))
RW631,1 & RW655    --> * RW670:  All x:Any + =(e1(d(e1(e1(e1(d(e(d(d(x)))))))))
                                               e(e(e(d(e(e(e(e(d(e(x))))))))))))
RW670,1 & RW622    --> * RW671:  All x:Any + =(d(e1(e1(e1(d(d(d(x)))))))) e(e(e(d(e(e(e(e(d(e(x))))))))))
RW671,1 & RW521    --> * RW672:  All x:Any + =(d(e1(e1(e(e(d(e(e(d(x)))))))))
                                               e(e(e(d(e(e(e(e(d(e(x))))))))))
```

```
RW672,1 & A16      --> * RW673:  All x:Any + =(d(e1(e(d(e(e(d(x)))))))) e(e(e(d(e(e(e(d(e(x))))))))))
RW673,1 & A16      --> * RW674:  All x:Any + =(d(d(e(e(d(x))))) e(e(e(d(e(e(e(d(e(x))))))))))
RW644,1 & RW655    --> * RW675:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) e(e(e(d(e(e(e(e(d(e(e(x))))))))))))
RW675,1 & RW655    --> * RW676:  All x:Any + =(d(e(e(d(d(e(e(x)))))))) e(e(e(e(d(e(e(e(d(e(e(x))))))))))))
RW454,1 & RW667    --> * RW677:  All x:Any + =(d(e(e(e(e(e(d(e(e(e(d(e(e(e(x)))))))))))))))) d(e1(x)))
RW621,1 & RW667    --> * RW680:  All x:Any + =(e(d(d(e(e(e(e(e(e(e(d(e(e(e(d(e(e(e(x)))))))))))))))))))
                                            d(d(e(e(x)))))
RW395,1 & RW667    --> * RW688:  All x:Any + =(e(e(e(e(e(e(d(e(e(e(d(e(e(e(d(x)))))))))))))))))
                                            d(e1(e1(e1(e1(e1(d(e1(x)))))))))
RW362,1 & RW521    --> * RW724:  All x:Any + =(e1(e1(e(e(d(e(e(d(x)))))))))
                                            d(e1(e1(d(e1(e1(e1(d(e(e(x))))))))))
RW724,1 & RW622    --> * RW725:  All x:Any + =(e1(e1(e(e(d(e(e(d(x)))))))) d(e1(d(e1(e1(e1(d(e(x))))))))
RW725,1 & RW622    --> * RW726:  All x:Any + =(e1(e1(e(e(d(e(e(d(x)))))))) d(d(e1(e1(e1(d(x)))))))
RW726,1 & RW633    --> * RW727:  All x:Any + =(e1(e1(e(e(d(e(e(d(x)))))))) d(e1(e1(d(e(e1(e1(d(x))))))))
RW727,1 & A16      --> * RW728:  All x:Any + =(e1(e(d(e(e(d(x)))))) d(e1(e1(d(e(e1(e1(d(x))))))))
RW728,1 & A16      --> * RW729:  All x:Any + =(d(e(e(d(x)))) d(e1(e1(d(e(e1(e1(d(x))))))))
RW729,1 & A15      --> * RW730:  All x:Any + =(d(e(e(d(x)))) d(e1(e1(d(e1(d(x)))))))
RW730,1 & RW466    --> * RW731:  All x:Any + =(d(e(e(d(x)))) d(e1(e1(e1(d(d(x)))))))
RW731,1 & RW521    --> * RW732:  All x:Any + =(d(e(e(d(x)))) d(e1(e1(e(e(d(e(e(x))))))))
RW732,1 & A16      --> * RW733:  All x:Any + =(d(e(e(d(x)))) d(e1(e(d(e(e(x)))))))
RW733,1 & A16      --> * RW734:  All x:Any + =(d(e(e(d(x)))) d(d(e(e(x)))))
RW592,1 & RW734    --> * RW737:  All x:Any + =(e(d(e(e(d(x)))))) d(e1(e1(e1(d(e(d(x))))))))
RW680,1 & RW734    --> * RW744:  All x:Any + =(e(d(e(e(d(e(e(e(e(e(e(d(e(e(e(d(e(e(e(x)))))))))))))))))))
                                            d(d(e(e(x)))))
RW744,1 & RW734    --> * RW745:  All x:Any + =(e(d(e(e(d(e(e(e(e(e(d(e(e(e(d(e(e(e(x)))))))))))))))))))
                                            d(e(e(d(x)))))
RW745,1 & RW677    --> * RW746:  All x:Any + =(e(d(e(e(d(e1(x)))))) d(e(e(d(x)))))
RW746,1 & RW453    --> * RW747:  All x:Any + =(e(d(e(d(x)))) d(e(e(d(x)))))
RW674,1 & RW734    --> * RW748:  All x:Any + =(d(e(e(d(d(x))))) e(e(e(d(e(e(e(d(e(x))))))))))
RW676,1 & RW734    --> * RW749:  All x:Any + =(d(e(e(d(e(e(d(x)))))))) e(e(e(e(d(e(e(e(d(e(e(x))))))))))))
RW591,1 & RW748    --> * RW756:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))
                                            d(e(e(e(e(d(e(e(e(d(e(e(e(d(x)))))))))))))))
RW756,1 & RW747    --> * RW757:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))
                                            d(e(e(e(e(d(e(e(e(e(d(e(d(x)))))))))))))))
RW757,1 & RW655    --> * RW758:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))
                                            d(e(e(e(e(e(d(e(e(e(e(d(e(d(x)))))))))))))))
RW758,1 & RW669    --> * RW759:  All x:Any + =(e1(d(e1(e1(e1(d(e(x)))))))) d(e(e(x))))
RW759,1 & RW622    --> * RW760:  All x:Any + =(d(e1(e1(e1(d(x))))) d(e(e(x))))
RW556,1 & RW748    --> * RW761:  All x:Any + =(d(e1(e1(e1(e1(x))))) e(e(e(e(d(e(e(e(e(d(e(d(x))))))))))))
RW761,1 & RW669    --> * RW762:  All x:Any + =(d(e1(e1(e1(e1(x))))) e(x))
RW688,1 & RW762    --> * RW770:  All x:Any + =(e(e(e(e(e(d(e(e(e(d(e(e(e(d(x)))))))))))))))
                                            e(e1(d(e1(x)))))
RW770,1 & A15      --> * RW771:  All x:Any + =(e(e(e(e(e(d(e(e(e(d(e(e(e(d(x)))))))))))))))) d(e1(x)))
RW737,1 & RW760    --> * RW777:  All x:Any + =(e(d(e(e(d(x))))) d(e(e(e(d(x))))))
RW777,1 & RW747    --> * RW778:  All x:Any + =(e(e(d(e(d(x))))) d(e(e(e(d(x))))))
RW543,1 & RW760    --> * RW781:  All x:Any + =(d(d(e1(e1(x)))) d(e(e(e(x)))))
RW781,1 & RW633    --> * RW782:  All x:Any + =(d(e1(e1(d(e(e1(x))))))) d(e(e(e(x)))))
RW782,1 & A15      --> * RW783:  All x:Any + =(d(e1(e1(d(x)))) d(e(e(e(x)))))
RW474,1 & RW783    --> * RW790:  - =(d(c_1) d(e(e(e(e1(e1(e1(c_1)))))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW790,1 & A15      --> * RW791:  - =(d(c_1) d(e(e(e1(e1(c_1))))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW791,1 & A15      --> * RW792:  - =(d(c_1) d(e(e1(c_1))))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW792,1 & A15      --> * RW793:  - =(d(c_1) d(c_1))  - =(d(c_2) e(e(e(e(e(c_2))))))
RW793,1 & A1,1     --> * R800:   - =(d(c_2) e(e(e(e(e(c_2))))))
RW669,1 & RW778    --> * RW819:  All x:Any + =(x e(e(e(e(e(d(e(d(e(d(x)))))))))))
RW749,1 & RW778    --> * RW831:  All x:Any + =(d(e(e(d(e(e(d(x)))))))) e(e(e(e(e(e(d(e(d(e(e(x))))))))))))
RW831,1 & RW747    --> * RW832:  All x:Any + =(e(d(e(d(e(e(d(x)))))))) e(e(e(e(e(e(d(e(d(e(e(x))))))))))))
RW832,1 & RW747    --> * RW833:  All x:Any + =(e(d(e(e(d(e(d(x)))))))) e(e(e(e(e(e(d(e(d(e(e(x))))))))))))
RW833,1 & RW747    --> * RW834:  All x:Any + =(e(e(d(e(d(e(d(x)))))))) e(e(e(e(e(e(d(e(d(e(e(x))))))))))))
RW771,1 & RW778    --> * RW835:  All x:Any + =(e(e(e(e(e(e(e(d(e(d(e(e(e(d(x))))))))))))))) d(e1(x)))
RW835,1 & RW778    --> * RW836:  All x:Any + =(e(e(e(e(e(e(e(d(e(e(e(d(e(d(x))))))))))))))) d(e1(x)))
RW836,1 & RW778    --> * RW837:  All x:Any + =(e(e(e(e(e(e(e(e(e(d(e(d(e(d(x))))))))))))))) d(e1(x)))
RW819,1 & RW834    --> * RW838:  All x:Any + =(x e(e(e(e(e(e(e(e(e(d(e(d(e(e(x))))))))))))))))
RW837,1 & RW834    --> * RW839:  All x:Any + =(e(e(e(e(e(e(e(e(e(e(e(e(d(e(d(e(e(x)))))))))))))))))))
                                            d(e1(x)))
RW839,1 & RW838    --> * RW842:  All x:Any + =(e(e(e(e(x)))) d(e1(x)))
```

```
RW453,1 & RW842    --> * RW866:  All x:Any + =(d(x) e(e(e(e(e(x))))))
RW866,1 & R800,1   --> * R867:   []
```

B.3.7 Z29

Z29 is a more complicated finite cyclic group. For this example we just present the input formulae because the proof is too long (686 inference steps) and in another format than the others displayed in this appendix. It was generated by the completion procedure of chapter 5. The result of the completion process are the 14 rules below.

This example can be regarded as a real challenge problem. It needed some days of computation time and about 180 MByte of swap space.

```
Set of Axiom Clauses Resulting from Normalization
=================================================

            A1:   All x:Any + =(x x)
            A2:   All x:Any + =(a(b(x)) c(x))
            A3:   All x:Any + =(b(c(x)) d(x))
            A4:   All x:Any + =(c(d(x)) e(x))
            A5:   All x:Any + =(d(e(x)) f(x))
            A6:   All x:Any + =(e(f(x)) g(x))
            A7:   All x:Any + =(f(g(x)) a(x))
            A8:   All x:Any + =(g(a(x)) b(x))
            A9:   All x:Any + =(a(a1(x)) x)
            A10:  All x:Any + =(b(b1(x)) x)
            A11:  All x:Any + =(c(c1(x)) x)
            A12:  All x:Any + =(d(d1(x)) x)
            A13:  All x:Any + =(e(e1(x)) x)
            A14:  All x:Any + =(f(f1(x)) x)
            A15:  All x:Any + =(g(g1(x)) x)
            A16:  All x:Any + =(a1(a(x)) x)
            A17:  All x:Any + =(b1(b(x)) x)
            A18:  All x:Any + =(c1(c(x)) x)
            A19:  All x:Any + =(d1(d(x)) x)
            A20:  All x:Any + =(e1(e(x)) x)
            A21:  All x:Any + =(f1(f(x)) x)
            A22:  All x:Any + =(g1(g(x)) x)

   Completion of z29.
a(b(x)) --> c(x)
b(c(x)) --> d(x)
c(d(x)) --> e(x)
d(e(x)) --> f(x)
e(f(x)) --> g(x)
f(g(x)) --> a(x)
g(a(x)) --> b(x)
d(d1(x)) --> x
d1(d(x)) --> x
e(e1(x)) --> x
e1(e(x)) --> x
f(f1(x)) --> x
f1(f(x)) --> x
g(g1(x)) --> x
g1(g(x)) --> x
c(9(x)) --> x
9(c(x)) --> x
b(b1(x)) --> x
b1(b(x)) --> x
a1(a(x)) --> x
a(a1(x)) --> x
New Rule (20): a(a1(x)) --> x
New Rule (19): a1(a(x)) --> x
New Rule (18): b1(b(x)) --> x
```

```
New Rule (17): b(b1(x)) --> x
New Rule (8): 9(c(x)) --> x
New Rule (7): c(9(x)) --> x
New Rule (16): g1(g(x)) --> x
New Rule (15): g(g1(x)) --> x
New Rule (14): f1(f(x)) --> x
New Rule (13): f(f1(x)) --> x
New Rule (12): e1(e(x)) --> x
New Rule (11): e(e1(x)) --> x
New Rule (10): d1(d(x)) --> x
New Rule (9): d(d1(x)) --> x
New Rule (6): g(a(x)) --> b(x)
New Rule (51): g(x) --> b(a1(x))
   derived from: 20 and 6
New Rule (16): g1(b(a1(x))) --> x
New Rule (54): g1(b(x)) --> a(x)
   derived from: 19 and 16
New Rule (56): g1(x) --> a(b1(x))
   derived from: 17 and 54
New Rule (5): f(b(a1(x))) --> a(x)
New Rule (4): e(f(x)) --> b(a1(x))
New Rule (3): f(x) --> d(e(x))
New Rule (14): f1(d(e(x))) --> x
New Rule (67): f1(d(x)) --> e1(x)
   derived from: 11 and 14
New Rule (69): f1(x) --> e1(d1(x))
   derived from: 9 and 67
New Rule (2): e(x) --> c(d(x))
New Rule (12): e1(c(d(x))) --> x
New Rule (73): e1(c(x)) --> d1(x)
   derived from: 9 and 12
New Rule (75): e1(x) --> d1(9(x))
   derived from: 7 and 73
New Rule (1): d(x) --> b(c(x))
New Rule (10): d1(b(c(x))) --> x
New Rule (79): d1(b(x)) --> 9(x)
   derived from: 7 and 10
New Rule (81): d1(x) --> 9(b1(x))
   derived from: 17 and 79
New Rule (0): c(x) --> a(b(x))
New Rule (8): 9(a(b(x))) --> x
New Rule (85): 9(a(x)) --> b1(x)
   derived from: 17 and 8
New Rule (87): 9(x) --> b1(a1(x))
   derived from: 20 and 85
New Rule (5): b(a(b(a(b(b(a(b(b(a1(x))))))))))) --> a(x)
New Rule (4): a(b(b(a(b(b(a(b(a(b(b(a(b(x))))))))))))) --> b(a1(x))
New Rule (62): a(b(b(a(b(a(x)))))) --> b(a1(b(a1(x))))
   derived from: 5 and 4
New Rule (105): a(b(b(a(b(x))))) --> b(a1(b(a1(a1(x)))))
   derived from: 20 and 62
New Rule (109): b(b(a(b(x)))) --> a1(b(a1(b(a1(a1(x))))))
   derived from: 19 and 105
New Rule (113): b1(a1(b(a1(b(a1(a1(x))))))) --> b(a(b(x)))
   derived from: 18 and 109
New Rule (116): b1(a1(b(a1(b(a1(x)))))) --> b(a(b(a(x))))
   derived from: 19 and 113
New Rule (119): b1(a1(b(a1(b(x))))) --> b(a(b(a(a(x)))))
   derived from: 19 and 116
New Rule (122): b1(a1(b(a1(x)))) --> b(a(b(a(a(b1(x))))))
   derived from: 17 and 119
New Rule (126): b1(a1(b(x))) --> b(a(b(a(a(b1(a(x)))))))
   derived from: 19 and 122
New Rule (129): b(a(b(a(a(b1(a(b1(x))))))))) --> b1(a1(x))
   derived from: 17 and 126
```

```
New Rule (137): b1(b1(a1(x))) --> a(b(a(a(b1(a(b1(x)))))))
   derived from: 18 and 129
New Rule (142): a(b(a(a(b1(a(b1(a(x))))))))) --> b1(b1(x))
   derived from: 19 and 137
New Rule (147): b(a(a(b1(a(b1(a(x)))))))) --> a1(b1(b1(x)))
   derived from: 19 and 142
New Rule (153): b1(a1(b1(b1(x)))) --> a(a(b1(a(b1(a(x))))))
   derived from: 18 and 147
New Rule (163): a(a(b1(a(b1(a(b(x)))))))) --> b1(a1(b1(x)))
   derived from: 18 and 153
New Rule (172): a(b1(a(b1(a(b(x)))))) --> a1(b1(a1(b1(x))))
   derived from: 19 and 163
New Rule (183): b1(a(b1(a(b(x))))) --> a1(a1(b1(a1(b1(x)))))
   derived from: 19 and 172
New Rule (194): b(a1(a1(b1(a1(b1(x)))))) --> a(b1(a(b(x))))
   derived from: 17 and 183
New Rule (204): a(b1(a(b(b(x))))) --> b(a1(a1(b1(a1(x)))))
   derived from: 18 and 194
New Rule (217): b1(a(b(b(x)))) --> a1(b(a1(a1(b1(a1(x))))))
   derived from: 19 and 204
New Rule (230): b(a1(b(a1(a1(b1(a1(x)))))))) --> a(b(b(x)))
   derived from: 17 and 217
New Rule (240): b(a1(b(a1(a1(b1(x)))))) --> a(b(b(a(x))))
   derived from: 19 and 230
New Rule (5): b(a(b(b(a1(b(a1(a1(b(a1(x))))))))))) --> a(x)
New Rule (263): b(a(b(b(a1(b(a1(a1(b(x)))))))))) --> a(a(x))
   derived from: 19 and 5
New Rule (277): b1(a(a(x))) --> a(b(b(a1(b(a1(a1(b(x)))))))))
   derived from: 18 and 263
New Rule (285): b1(a(x)) --> a(b(b(a1(b(a1(a1(b(a1(x)))))))))
   derived from: 20 and 277
New Rule (291): b1(x) --> a(b(b(a1(b(a1(a1(b(a1(a1(x))))))))))
   derived from: 20 and 285
New Rule (18): a(b(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))) --> x
New Rule (298): b(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))) --> a1(x)
   derived from: 19 and 18
New Rule (4): a(b(b(b(a1(b(a1(a1(b(a1(b(a1(a1(x))))))))))))) --> b(a1(x))
New Rule (308): a(b(b(b(a1(b(a1(a1(b(a1(b(a1(x))))))))))))) --> b(x)
   derived from: 19 and 4
New Rule (313): b(b(b(a1(b(a1(a1(b(a1(b(a1(x))))))))))) --> a1(b(x))
   derived from: 19 and 308
New Rule (318): b(b(b(a1(b(a1(a1(b(a1(b(x))))))))))) --> a1(b(a(x)))
   derived from: 19 and 313
New Rule (325): a1(b(a(b(a(b(x)))))) --> b(a1(a1(b(a1(a1(x))))))
   derived from: 109 and 318
New Rule (336): b(a(b(a(b(x))))) --> a(b(a1(a1(b(a1(a1(x)))))))
   derived from: 20 and 325
New Rule (348): b(a(b(a1(a1(b(a1(a1(x))))))))) --> a1(b(a1(b(a1(b(x))))))
   derived from: 109 and 336
New Rule (358): b(a(b(a1(a1(b(a1(x)))))))) --> a1(b(a1(b(a1(b(a(x)))))))
   derived from: 19 and 348
New Rule (367): a1(b(a1(b(a1(b(a(a(x))))))))) --> b(a(b(a1(a1(b(x))))))
   derived from: 19 and 358
New Rule (376): b(a1(b(a1(b(a(a(x)))))))) --> a(b(a(b(a1(a1(b(x)))))))
   derived from: 20 and 367
New Rule (274): a1(b(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))) --> b(a(a(x)))
   derived from: 109 and 263
New Rule (402): b(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))) --> a(b(a(a(x))))
   derived from: 20 and 274
New Rule (362): a1(b(a(a(b(a1(a1(b(a1(x))))))))) --> b(a1(a1(b(a1(b(a(x)))))))
   derived from: 318 and 358
New Rule (433): b(a(a(b(a1(a1(b(a1(x))))))))) --> a(b(a1(a1(b(a1(b(a(x))))))))
   derived from: 20 and 362
New Rule (444): a(b(a1(a1(b(a1(b(a(a(x))))))))) --> b(a(a(b(a1(a1(b(x)))))))
   derived from: 19 and 433
```

```
New Rule (453): b(a1(a1(b(a1(b(a(a(x)))))))) --> a1(b(a(a(b(a1(a1(b(x)))))))
   derived from: 19 and 444
New Rule (265): a(a(a(b(b(a1(b(a1(a1(b(x))))))))))) --> b(a(b(b(a1(b(x))))))
   derived from: 263 and 263
New Rule (488): b(a(b(b(a(x))))) --> a(a(a(b(b(a1(b(x)))))))
   derived from: 263 and 265
New Rule (508): a(a(a(b(b(a1(b(a1(x)))))))) --> b(a(b(b(x))))
   derived from: 20 and 488
New Rule (522): a(a(b(b(a1(b(a1(x))))))) --> a1(b(a(b(b(x)))))
   derived from: 19 and 508
New Rule (536): a(b(b(a1(b(a1(x)))))) --> a1(a1(b(a(b(b(x))))))
   derived from: 19 and 522
New Rule (547): a1(a1(b(a1(a1(b(a(b(b(a1(b(x))))))))))) --> x
   derived from: 298 and 536
New Rule (574): a1(b(a1(a1(b(a(b(b(a1(b(x))))))))))) --> a(x)
   derived from: 20 and 547
New Rule (602): b(a1(a1(b(a(b(b(a1(b(x)))))))))) --> a(a(x))
   derived from: 20 and 574
New Rule (605): b(a1(a1(b(a1(a1(b(a(b(b(x))))))))))) --> a(x)
   derived from: 536 and 602
New Rule (657): b(b(a1(b(a1(x))))) --> a1(a1(a1(b(a(b(b(x)))))))
   derived from: 298 and 605
New Rule (505): a(a(a(b(b(a1(b(b(x)))))))) --> a1(a1(a1(b(a(b(b(a1(x))))))))
   derived from: 109 and 488
New Rule (318): b(a1(a1(a1(b(a1(a1(b(a(b(b(b(x))))))))))))) --> a1(b(a(x)))
New Rule (703): b(b(a1(a1(b(a(x)))))) --> a1(a1(b(a1(a1(b(a1(a1(x))))))))
   derived from: 657 and 318
New Rule (739): b(b(a1(a1(b(x))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(x))))))))))
   derived from: 20 and 703
New Rule (752): a1(a1(b(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))) --> b(a(a(x)))
   derived from: 602 and 739
New Rule (773): b(b(a1(b(b(a(a(x)))))))) --> a1(a1(a1(b(b(a1(b(x)))))))
   derived from: 657 and 752
New Rule (805): b(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))) --> a(a(b(a(a(x)))))
   derived from: 602 and 773
New Rule (836): b(a1(b(a1(a1(b(a(b(a(a(x))))))))))) --> a(b(b(b(a1(b(x))))))
   derived from: 402 and 805
New Rule (811): b(b(a1(b(b(a(x)))))) --> a1(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))
   derived from: 20 and 773
New Rule (890): b(b(a1(b(b(x))))) --> a1(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))
   derived from: 20 and 811
New Rule (613): a(b(a(a(a(b(b(a1(b(x))))))))) --> b(a1(b(a1(a1(b(a(x)))))))
   derived from: 402 and 602
New Rule (953): b(a(a(a(b(b(a1(b(x))))))))) --> a1(b(a1(b(a1(a1(b(a(x))))))))
   derived from: 19 and 613
New Rule (111): b(a1(a1(a1(b(a(b(b(a1(x))))))))) --> a1(b(a1(b(a1(a1(b(a(b(x))))))))))
   derived from: 109 and 109
New Rule (1012): b(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))) --> a1(b(a(a(a(x)))))
   derived from: 376 and 111
New Rule (992): a1(b(a1(b(a1(b(a1(a1(b(a1(a1(b(a(b(x))))))))))))))) --> a1(b(a1(x)))
   derived from: 953 and 111
New Rule (1083): a(b(a1(b(a1(a1(b(a1(a1(b(a(b(x))))))))))))) --> a(x)
   derived from: 602 and 992
New Rule (1134): b(a1(b(a1(a1(b(a1(a1(b(a(b(x))))))))))) --> x
   derived from: 19 and 1083
New Rule (910): b(a1(a1(b(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))) --> a(a(b(x)))
   derived from: 602 and 890
New Rule (1219): b(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))))) --> a(a(b(a(x))))
   derived from: 19 and 910
New Rule (862): b(a1(b(a1(b(b(b(a1(b(x))))))))) --> a(b(a(a(a(b(a(a(x))))))))
   derived from: 402 and 836
New Rule (666): a1(a1(a1(b(a(b(b(a1(b(a(b(b(a1(b(x))))))))))))))) --> b(b(a(x)))
   derived from: 602 and 657
New Rule (1355): a1(a1(b(a(b(b(a1(b(a(b(b(a1(b(x))))))))))))))) --> a(b(b(a(x))))
   derived from: 20 and 666
```

```
New Rule (1398): a1(b(a(b(b(a1(b(a(b(b(a1(b(x))))))))))) --> a(a(b(b(a(x)))))
   derived from: 20 and 1355
New Rule (1447): b(a(b(b(a1(b(a(b(b(a1(b(x))))))))))) --> a(a(a(b(b(a(x))))))
   derived from: 20 and 1398
New Rule (648): a1(b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))))))) --> b(a(b(a1(x))))
   derived from: 358 and 605
New Rule (380): b(a1(a1(a1(b(a(b(a1(b(a1(b(a1(a1(a1(a1(b(x))))))))))))))) --> a1(b(b(a(a(x)))))
   derived from: 318 and 376
New Rule (129): a1(a1(a1(b(a1(a1(b(a(b(b(b(a1(a1(a1(b(a1(a1(b(a(b(b(a1(x))))))))))))))))))))
                --> a1(a1(b(a1(a1(b(a(b(b(a1(a1(x))))))))))
New Rule (147): a1(a1(a1(b(a1(a1(b(a(b(b(a1(a1(a1(b(a1(a1(b(a(b(b(a1(x))))))))))))))))))))
                --> a1(b(a1(b(a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(b(a(b(b(x))))))))))))))))))
New Rule (183): a1(a1(a1(a1(b(a1(a1(b(a(b(b(a1(a1(a1(a1(b(a1(a1(b(a(b(b(a1(x))))))))))))))))))))))
                --> a1(a1(b(a1(b(a1(a1(b(a1(a1(x))))))))))
New Rule (217): a1(a1(b(a1(a1(b(a(b(b(b(x))))))))))
                --> a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(b(a1(a1(x))))))))))))))))
New Rule (275): a1(b(a1(b(a1(a1(a1(a1(b(a(b(b(a1(b(x))))))))))))) --> b(b(a(a(a(x)))))
   derived from: 109 and 2
New Rule (1746): b(b(a(a(a(a(b(b(a1(b(x))))))))))) --> a1(b(a1(b(a1(b(b(a(x))))))))
   derived from: 1447 and 275
New Rule (1455): b(a1(a1(b(a1(a1(b(b(a(x))))))))) --> a(b(a1(b(a1(a1(b(a1(b(x))))))))
   derived from: 1219 and 1447
New Rule (1896): a(b(a1(b(a1(a1(b(a1(b(a1(x))))))))))) --> b(a1(a1(b(a1(a1(b(b(x))))))))
   derived from: 20 and 1455
New Rule (1938): b(a1(b(a1(a1(b(a1(b(a1(x))))))))) --> a1(b(a1(a1(b(a1(a1(b(b(x))))))))
   derived from: 19 and 1896
New Rule (1970): b(a1(b(a1(a1(b(a(b(a(x))))))))) --> a(b(a1(a1(a1(b(a(b(b(x))))))))
   derived from: 1219 and 1938
New Rule (1225): b(a1(a1(b(a1(a1(b(a(b(a(x))))))))))) --> a1(b(a1(a1(b(a(b(b(x))))))))
   derived from: 1012 and 1219
New Rule (2092): a1(b(a1(a1(b(a(b(b(a1(x))))))))) --> b(a1(a1(b(a1(a1(b(a(b(x))))))))
   derived from: 20 and 1225
New Rule (2142): b(a1(a1(b(a(b(b(a1(x))))))))) --> a(b(a1(a1(b(a1(a1(b(a(b(x))))))))))
   derived from: 20 and 2092
New Rule (499): b(a(a(a(a(b(b(a1(b(x))))))))) --> a(b(a1(a1(b(a1(a1(b(a(x))))))))
   derived from: 336 and 488
New Rule (498): a1(a1(a1(b(a(b(b(b(x))))))))) --> a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(x))))))))))))
   derived from: 336 and 488
New Rule (461): a1(b(a1(b(a1(b(a(b(a(a(x))))))))))) --> b(b(a(a(b(a1(a1(b(x))))))))
   derived from: 358 and 453
New Rule (2321): b(a1(b(a1(b(a(b(a(a(x))))))))) --> a(b(b(a(a(b(a1(a1(b(x))))))))
   derived from: 20 and 461
New Rule (365): b(a1(b(a1(b(a1(b(a(x))))))))) --> a1(b(a1(b(a1(a1(a1(a1(b(a1(x))))))))))
   derived from: 109 and 358
New Rule (2403): b(a1(b(a1(b(a1(b(x))))))) --> a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(x))))))))))
   derived from: 20 and 365
New Rule (2147): b(a1(a1(b(a(b(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))))))) --> b(a1(b(a1(x))))
   derived from: 1938 and 2142
New Rule (2535): a1(b(a(b(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))))) --> b(a1(x))
   derived from: 605 and 2147
New Rule (2601): b(a(b(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))))) --> a(b(a1(x)))
   derived from: 20 and 2535
New Rule (2662): b(a(b(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))))))) --> a(b(b(x)))
   derived from: 109 and 2601
New Rule (2717): b(a(b(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(x))))))))))))))) --> a(b(b(a(x))))
   derived from: 19 and 2662
New Rule (2780): b(a(b(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(x))))))))))))))) --> a(b(b(a(a(x)))))
   derived from: 19 and 2717
New Rule (2626): a(b(a1(a1(b(a1(b(b(b(a1(b(x))))))))))) --> b(a(a(a(a(b(a(a(x))))))))
   derived from: 862 and 2601
New Rule (2907): b(a1(a1(b(a1(b(b(b(a1(b(x))))))))))) --> a1(b(a(a(a(a(b(a(a(x))))))))
   derived from: 19 and 2626
New Rule (2623): a1(b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(a(b(a1(x))))))))))))))))) --> b(b(x))
   derived from: 648 and 2601
New Rule (3038): b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(a(b(a1(x))))))))))))))))) --> a(b(b(x)))
```

```
   derived from: 20 and 2623
New Rule (3107): b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(a(b(x)))))))))))))) --> a(b(b(a(x))))
   derived from: 19 and 3038
New Rule (3131): b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(b(x)))))))))))))) --> a(b(b(a(a(x)))))
   derived from: 1225 and 3107
New Rule (3264): b(a1(b(a1(a1(a1(a1(b(a(b(b(a1(b(x)))))))))))) --> a(b(b(a(a(a(x))))))
   derived from: 488 and 3131
New Rule (2072): a1(b(a1(b(a1(b(a1(a1(a1(b(a(b(b(x)))))))))))) --> b(a(a(a(b(a(x))))))
   derived from: 953 and 1225
New Rule (3438): b(a1(b(a1(b(a1(a1(a1(b(a(b(b(x)))))))))))) --> a(b(a(a(a(b(a(x)))))))
   derived from: 20 and 2072
New Rule (1940): b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(b(x)))))))))))))) --> a(b(a(b(a1(x)))))
   derived from: 1938 and 1938
New Rule (1241): b(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(x)))))))))))))) --> a(a(a(b(x))))
   derived from: 318 and 1219
New Rule (3659): b(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(x)))))))))))))) --> a(a(a(b(a1(x)))))
   derived from: 20 and 1241
New Rule (1174): b(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(b(a(x)))))))))))))) --> a1(a1(b(a1(x))))
   derived from: 358 and 1134
New Rule (3808): b(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(b(x)))))))))))))) --> a1(a1(b(a1(a1(x)))))
   derived from: 20 and 1174
New Rule (660): b(a1(a1(b(a1(a1(b(a(b(a1(b(a1(b(a1(a1(x)))))))))))))) --> a(b(a(b(x))))
   derived from: 109 and 605
New Rule (3982): b(a1(a1(b(a1(a1(b(a(b(a1(b(a1(b(a1(x)))))))))))))) --> a(b(a(b(a(x)))))
   derived from: 19 and 660
New Rule (4068): b(a1(a1(b(a1(a1(b(a(b(a1(b(a1(b(x)))))))))))))) --> a(b(a(b(a(a(x))))))
   derived from: 19 and 3982
New Rule (641): a(b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(b(x)))))))))))))))) --> b(a(a(b(a1(x)))))
   derived from: 433 and 605
New Rule (619): b(a1(a1(a1(b(a1(b(a1(b(b(a1(a1(a1(a1(a1(x)))))))))))))) --> a(a(a(b(a(b(x))))))
   derived from: 336 and 0
New Rule (503): a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(x)))))))))))))))) --> b(a(b(a(a(x)))))
   derived from: 2 and 488
New Rule (346): a(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(x)))))))))))))))) --> a(a(a(b(a(b(x))))))
   derived from: 2 and 336
New Rule (345): a(b(a1(a1(b(a1(a1(a1(a1(b(a(b(b(a1(b(x)))))))))))))))) --> b(a(b(a(a(a(x))))))
   derived from: 2 and 336
New Rule (333): b(a1(a1(b(a1(a1(a1(a1(b(a(b(b(a1(b(x)))))))))))))) --> a1(b(a(b(a(a(a(x)))))))
   derived from: 2 and 325
New Rule (326): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(x)))))))))))))))))) --> b(b(b(a(x))))
   derived from: 109 and 318
New Rule (273): b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(b(a1(a1(x)))))))))))))))) --> a(a(b(a(b(x)))))
   derived from: 109 and 2
New Rule (4674): a1(b(a1(a1(b(a(b(b(b(x)))))))))) --> b(a1(a1(b(a1(b(a1(a1(b(a1(a1(x)))))))))))
   derived from: 1012 and 273
New Rule (4781): b(a1(a1(b(a(b(b(b(x)))))))) --> a(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(x)))))))))))
   derived from: 20 and 4674
New Rule (4850): b(a(b(b(a1(b(a(b(a(a(x)))))))))) --> a(a(a(b(a(b(b(b(a1(b(x)))))))))
   derived from: 805 and 4781
New Rule (4129): b(a1(a1(b(a1(a1(b(a(b(a1(x)))))))))) --> a(b(a1(b(a1(a1(b(a1(a1(b(x)))))))))
   derived from: 1134 and 4068
New Rule (5011): a(b(a1(b(a1(a1(b(a1(a1(b(a(x)))))))))))) --> b(a1(a1(b(a1(a1(b(a(b(x)))))))))
   derived from: 19 and 4129
New Rule (5045): b(a1(b(a1(a1(b(a1(a1(b(a(x)))))))))) --> a1(b(a1(a1(b(a1(a1(b(a(b(x)))))))))
   derived from: 19 and 5011
New Rule (5000): a(b(a1(a1(b(a(a(a(a(b(b(x)))))))))))) --> b(a1(a1(b(a1(a1(b(a(a(x)))))))))
   derived from: 605 and 4129
New Rule (5180): b(a1(a1(b(a(a(a(a(b(b(x)))))))))) --> a1(b(a1(a1(b(a1(a1(b(a(a(x)))))))))
   derived from: 19 and 5000
New Rule (3716): b(a1(a1(b(a1(a1(b(a(a(b(a1(x)))))))))))) --> a1(b(a1(a1(b(a(b(a1(b(x)))))))))
   derived from: 1012 and 3659
New Rule (5282): a(a(a(a(a(a(b(b(a1(b(a(b(a1(x)))))))))))))) --> b(a(b(a1(b(x)))))
   derived from: 4781 and 3716
New Rule (5411): a(a(a(a(a(b(b(a1(b(a(b(a1(x)))))))))))) --> a1(b(a(b(a1(b(x))))))
   derived from: 19 and 5282
```

```
New Rule (5466): a(a(a(a(b(b(a1(b(a(b(a1(x)))))))))) --> a1(a1(b(a(b(a1(b(x)))))))
   derived from: 19 and 5411
New Rule (5523): a(a(a(b(b(a1(b(a(b(a1(x)))))))))) --> a1(a1(a1(b(a(b(a1(b(x)))))))
   derived from: 19 and 5466
New Rule (5581): a(a(b(b(a1(b(a(b(a1(x)))))))) --> a1(a1(a1(a1(b(a(b(a1(b(x)))))))))
   derived from: 19 and 5523
New Rule (5639): a(b(b(a1(b(a(b(a1(x)))))))) --> a1(a1(a1(a1(a1(b(a(b(a1(b(x)))))))))))
   derived from: 19 and 5581
New Rule (5705): b(b(a1(b(a(b(a1(x)))))))) --> a1(a1(a1(a1(a1(a1(b(a(b(a1(b(x)))))))))))
   derived from: 19 and 5639
New Rule (5797): b(b(a1(b(a(b(x)))))) --> a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(x)))))))))))
   derived from: 19 and 5705
New Rule (5873): b(a1(a1(a1(b(a(b(a1(b(a(x)))))))))) --> a1(b(a1(b(a1(a1(b(a(a(b(x)))))))))
   derived from: 953 and 5797
New Rule (5949): a1(b(a1(b(a1(a1(b(a(a(b(a1(x)))))))))) --> b(a1(a1(a1(b(a(b(a1(b(x)))))))))
   derived from: 20 and 5873
New Rule (6026): b(a1(b(a1(a1(b(a(a(b(a1(x)))))))))) --> a(b(a1(a1(a1(b(a(b(a1(b(x)))))))))
   derived from: 20 and 5949
New Rule (5917): b(a1(b(a1(b(b(a(a(b(x)))))))) --> a1(b(a1(a1(b(a1(a1(b(a1(b(a(x)))))))))))
   derived from: 3107 and 5873
New Rule (5852): b(a1(a1(b(a(b(a1(b(a(x)))))))) --> a(b(a1(a1(b(a1(a1(b(a(a(b(x)))))))))))
   derived from: 499 and 5797
New Rule (5311): b(a(b(a1(a1(b(a(b(b(x)))))))) --> a(b(a1(a1(b(a1(a1(b(a1(b(a1(x)))))))))))
   derived from: 2780 and 3716
New Rule (2606): a(b(a1(a1(b(a1(b(a(b(a(a(x)))))))))) --> b(a(b(a(a(b(a1(a1(b(x)))))))))
   derived from: 2321 and 2601
New Rule (6411): b(a1(a1(b(a1(b(a(b(a(a(x)))))))))) --> a1(b(a(b(a(a(b(a1(a1(b(x)))))))))
   derived from: 19 and 2606
New Rule (2295): a1(a1(b(a(b(b(b(x)))))))) --> a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(x)))))))))))))
   derived from: 20 and 498
New Rule (6548): a1(b(a(b(b(b(x)))))) --> a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(x)))))))))))))
   derived from: 20 and 2295
New Rule (6642): b(a(b(b(b(x))))) --> a(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(x)))))))))))))))
   derived from: 20 and 6548
New Rule (1890): a1(b(a1(b(a1(b(b(b(a(x)))))))) --> b(a(a(b(a1(b(a1(a1(b(a1(b(x)))))))))))
   derived from: 358 and 1455
New Rule (6797): b(a1(b(a1(b(b(b(a(x)))))))) --> a(b(a(a(b(a1(b(a1(a1(b(a1(b(x)))))))))))
   derived from: 20 and 1890
New Rule (855): a1(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(a(x)))))))))))))))))
                --> a(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(b(x)))))))))))))))))
   derived from: 657 and 0
New Rule (6086): b(a1(b(a1(b(a1(a1(a1(b(a(b(a1(b(x)))))))))))) --> a(b(a(a(a(a(b(a1(x)))))))
   derived from: 1938 and 6026
New Rule (5109): a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(x)))))))))))))))) --> a(b(a(b(a(x)))))
   derived from: 433 and 5045
New Rule (7111): b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(x)))))))))))))))) --> a(a(b(a(b(a(x))))))
   derived from: 20 and 5109
New Rule (7207): b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(x)))))))))))))) --> a(a(b(a(b(a(a(x)))))))
   derived from: 19 and 7111
New Rule (5094): b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(a(b(x)))))))))))))))) --> a1(b(a(b(a(x)))))
   derived from: 1012 and 5045
New Rule (7393): b(a1(a1(a1(b(a1(a1(b(a(b(a1(b(a1(b(x)))))))))))))) --> a1(b(a(a(b(a(a(x)))))))
   derived from: 1012 and 5094
New Rule (7381): b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(b(x)))))))))))))))) --> a1(b(a(b(a(a(x))))))
   derived from: 1225 and 5094
New Rule (4273): b(a1(a1(a1(b(a1(b(a1(b(b(a1(a1(a1(a1(x)))))))))))))) --> a(a(a(b(a(b(a(x)))))))
   derived from: 19 and 619
New Rule (7721): b(a1(a1(a1(b(a1(b(a1(b(b(a1(a1(a1(x)))))))))))))) --> a(a(a(b(a(b(a(a(x)))))))
   derived from: 19 and 4273
New Rule (7779): b(a1(a1(a1(b(a1(b(a1(b(b(a1(a1(x)))))))))))) --> a(a(a(b(a(b(a(a(x)))))))
   derived from: 19 and 7721
New Rule (7856): b(a1(a1(a1(b(a1(b(a1(b(b(a1(x)))))))))) --> a(a(a(b(a(b(a(a(a(x)))))))))
   derived from: 19 and 7779
New Rule (7952): b(a1(a1(a1(b(a1(b(a1(b(b(x)))))))))) --> a(a(a(b(a(b(a(a(a(a(x)))))))))))
   derived from: 19 and 7856
```

```
New Rule (4228): b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(b(b(x)))))))))))))) --> a1(b(a(a(b(a1(x))))))
   derived from: 19 and 641
New Rule (3455): a(b(a1(a1(b(a1(b(a1(a1(a1(b(a(b(b(x)))))))))))))) --> b(a(a(a(a(b(a(x)))))))
   derived from: 2601 and 3438
New Rule (8270): b(a1(a1(b(a1(b(a1(a1(a1(b(a(b(b(x)))))))))))) --> a1(b(a(a(a(a(b(a(x))))))))
   derived from: 19 and 3455
New Rule (2978): a1(b(a1(b(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(b(x)))))))))))))))))))
                 --> b(b(a(a(a(b(a(a(x))))))))
   derived from: 358 and 2907
New Rule (2416): b(a1(a1(b(a(b(a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(x)))))))))))))))) --> a(b(a1(b(x))))
   derived from: 2142 and 2403
New Rule (8584): b(a1(a1(b(a(b(a1(b(a1(b(a1(a1(a1(a1(b(a1(x)))))))))))))))) --> a(b(a1(b(a(x)))))
   derived from: 19 and 2416
New Rule (8693): b(a1(a1(b(a(b(a1(b(a1(b(a1(a1(a1(a1(b(x)))))))))))))))) --> a(b(a1(b(a(a(x))))))
   derived from: 19 and 8584
New Rule (1942): b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(x)))))))))))))))))) --> a(b(b(b(a(x)))))
   derived from: 1455 and 1938
New Rule (1343): a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(b(a(b(b(a1(b(x)))))))))))))))))))
                 --> b(b(a(a(b(b(a(x))))))
   derived from: 488 and 0
New Rule (1297): a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x)))))))))))))))
                 --> b(a(b(a(a(a(b(a(a(x))))))))
   derived from: 657 and 862
New Rule (1243): b(a1(a1(b(a1(a1(a1(a1(a1(b(a1(a1(b(a(b(b(x)))))))))))))))) --> a(a(b(b(a1(x)))))
   derived from: 657 and 1219
New Rule (1216): a1(b(a1(b(a1(b(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x)))))))))))))))) --> b(a(b(b(x))))
   derived from: 358 and 0
New Rule (9356): b(a1(b(a1(b(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x)))))))))))))))) --> a(b(a(b(b(x)))))
   derived from: 20 and 1216
New Rule (9476): b(a1(b(a1(b(b(a1(a1(a1(a1(a1(b(a(b(a(x)))))))))))))))) --> a(a(a(a(b(b(x))))))
   derived from: 605 and 9356
New Rule (9571): b(a1(b(a1(b(b(a1(a1(a1(a1(a1(b(a(b(x)))))))))))))) --> a(a(a(a(b(b(a1(x)))))))
   derived from: 20 and 9476
New Rule (972): a1(b(a1(b(a1(a1(b(a1(b(a(b(b(a1(b(x)))))))))))))) --> b(a(a(a(b(b(a(x)))))))
   derived from: 0 and 953
New Rule (9843): b(a1(b(a1(a1(b(a1(b(a(b(b(a1(b(x)))))))))))))) --> a(b(a(a(a(b(b(a(x))))))))
   derived from: 20 and 972
New Rule (817): a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(b(a1(b(x)))))))))))))) --> b(a(a(b(a(a(x))))))
   derived from: 739 and 805
New Rule (10110): a1(b(a1(a1(b(a1(a1(a1(a1(a1(b(b(a1(b(x)))))))))))))) --> a(b(a(a(b(a(a(x))))))
   derived from: 20 and 817
New Rule (10239): b(a1(a1(b(a1(a1(a1(a1(a1(b(b(a1(b(x)))))))))))))) --> a(a(b(a(a(b(a(a(x))))))))
   derived from: 20 and 10110
New Rule (637): a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(b(x)))))))))))))))) --> a(a(b(b(a(x)))))
   derived from: 488 and 605
New Rule (10516): b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(b(x)))))))))))))))) --> a(a(a(b(b(a(x))))))
   derived from: 20 and 637
New Rule (10295): a(a(a(b(a1(a1(a1(b(a1(a1(b(b(x)))))))))))) --> b(a1(a1(b(a1(a1(a1(a1(b(a1(x))))))))))
   derived from: 3716 and 10239
New Rule (10753): a(a(b(a1(a1(a1(b(a1(a1(b(b(x)))))))))))) --> a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(x))))))))))))
   derived from: 19 and 10295
New Rule (10846): a(b(a1(a1(a1(b(a1(a1(b(b(x)))))))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(x))))))))))))
   derived from: 19 and 10753
New Rule (10909): a(b(a1(a1(a1(b(a1(a1(b(a(b(a1(a1(a1(b(a(b(b(x)))))))))))))))))) --> b(b(a1(x)))
   derived from: 1970 and 10846
New Rule (11039): b(a1(a1(a1(b(a1(a1(b(a(b(a1(a1(a1(b(a(b(b(x)))))))))))))))))) --> a1(b(b(a1(x))))
   derived from: 19 and 10909
New Rule (10941): b(a1(a1(a1(b(a1(a1(b(b(x)))))))))
                  --> a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(x))))))))))))))
   derived from: 19 and 10846
New Rule (9446): a(b(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x)))))))))))))) --> b(a1(b(a1(a1(b(a1(b(b(x)))))))))
   derived from: 2142 and 0
New Rule (11398): b(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x)))))))))))) --> a1(b(a1(b(a1(a1(b(a1(b(b(x))))))))))
   derived from: 19 and 9446
New Rule (11528): a1(b(a1(b(a1(a1(b(a1(b(b(a(x)))))))))) --> b(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))
```

```
   derived from: 19 and 11398
New Rule (11588): b(a1(b(a1(a1(b(a1(b(b(a(x)))))))))) --> a(b(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))
   derived from: 20 and 11528
New Rule (8053): a(a(a(b(a(b(a(a(a(a(a(b(b(a(x)))))))))))))) --> b(a1(a1(b(a(a(a(x)))))))
   derived from: 488 and 7952
New Rule (11717): a(a(a(b(a(b(a(a(a(a(a(b(b(x)))))))))))))) --> b(a1(a1(b(a(a(x))))))
   derived from: 20 and 8053
New Rule (11814): a(a(b(a(b(a(a(a(a(a(b(b(x)))))))))))) --> a1(b(a1(a1(b(a(a(x)))))))
   derived from: 19 and 11717
New Rule (11911): a(b(a(b(a(a(a(a(a(b(b(x)))))))))))) --> a1(a1(b(a1(a1(b(a(a(x)))))))
   derived from: 19 and 11814
New Rule (12012): b(a(b(a(a(a(a(a(b(b(x)))))))))) --> a1(a1(a1(b(a1(a1(b(a(a(x)))))))))
   derived from: 19 and 11911
New Rule (12155): a1(b(a1(b(a(a(a(a(b(b(x)))))))))) --> b(a1(a1(a1(b(a1(a1(b(a(a(x)))))))))
   derived from: 109 and 12012
New Rule (12280): b(a1(b(a(a(a(a(b(b(x)))))))))) --> a(b(a1(a1(a1(b(a1(a1(b(a(a(x)))))))))))
   derived from: 20 and 12155
New Rule (12140): b(a(b(a(a(a(b(a(b(b(x)))))))))) --> a1(a1(a1(b(a1(a1(b(a(b(a1(x)))))))))
   derived from: 657 and 12012
New Rule (12537): b(a(b(a(a(a(a(b(a1(a1(b(a1(a1(a1(x))))))))))))) --> a1(a1(a1(b(a1(a1(b(b(x)))))))))
   derived from: 2601 and 12140
New Rule (12687): b(a(b(a(a(a(a(b(a1(a1(b(a1(a1(x))))))))))))) --> a1(a1(a1(b(a1(a1(b(b(a(x)))))))))
   derived from: 19 and 12537
New Rule (12812): b(a(b(a(a(a(a(b(a1(a1(b(a1(x)))))))))))) --> a1(a1(a1(b(a1(a1(b(b(a(a(x)))))))))
   derived from: 19 and 12687
New Rule (12963): b(a(b(a(a(a(a(b(a1(a1(b(x)))))))))))) --> a1(a1(a1(b(a1(a1(b(b(a(a(a(x)))))))))))
   derived from: 19 and 12812
New Rule (12434): a(b(a(b(a1(a1(b(a(a(b(b(x)))))))))))) --> b(b(a1(a1(a1(b(a1(a1(b(a(a(x)))))))))))
   derived from: 376 and 12280
New Rule (13238): b(a(b(a1(a1(b(a(a(b(b(x)))))))))) --> a1(b(b(a1(a1(a1(b(a1(a1(b(a(a(x)))))))))))
   derived from: 19 and 12434
New Rule (6831): a(b(a1(a1(b(a1(b(b(b(a(x)))))))))) --> b(a(a(a(b(a1(b(a1(a1(b(a1(b(x)))))))))))
   derived from: 2601 and 6797
New Rule (13433): b(a1(a1(b(a1(b(b(b(a(x)))))))))) --> a1(b(a(a(a(b(a1(b(a1(a1(b(a1(b(x)))))))))))))
   derived from: 19 and 6831
New Rule (6331): b(a(b(a1(b(a1(a1(b(a1(a1(a1(x)))))))))))) --> a(b(a1(a1(b(a1(a1(b(a1(a1(b(x)))))))))))
   derived from: 2601 and 5311
New Rule (13619): b(a(b(a1(b(a1(a1(b(a1(a1(x)))))))))) --> a(b(a1(a1(b(a1(a1(b(a1(a1(b(a(x)))))))))))
   derived from: 19 and 6331
New Rule (13747): b(a(b(a1(b(a1(a1(b(a1(x)))))))))) --> a(b(a1(a1(b(a1(a1(b(a1(a1(b(a(a(x)))))))))))))
   derived from: 19 and 13619
New Rule (13905): a(b(a1(a1(b(a1(a1(b(a1(a1(b(a(a(a(x)))))))))))))) --> b(a(b(a1(b(a1(a1(b(x)))))))))
   derived from: 19 and 13747
New Rule (13940): b(a1(a1(b(a1(a1(b(a1(a1(b(a(a(a(x)))))))))))))) --> a1(b(a(b(a1(b(a1(a1(b(x)))))))))
   derived from: 19 and 13905
New Rule (13797): b(a(b(a1(a1(b(a(b(a(a(x)))))))))) --> a(b(a1(a1(a1(b(a(a(a(a(b(b(x)))))))))))))
   derived from: 7381 and 13747
New Rule (14094): a(b(a1(a1(a1(b(a(a(a(a(b(b(a1(x)))))))))))))) --> b(a(b(a1(a1(b(a(b(a(x)))))))))
   derived from: 20 and 13797
New Rule (14189): b(a1(a1(a1(b(a(a(a(a(b(b(a1(x)))))))))))) --> a1(b(a(b(a1(a1(b(a(b(a(x)))))))))))
   derived from: 19 and 14094
New Rule (14093): b(b(a(a(b(a1(a1(a1(b(a(a(a(a(b(b(x)))))))))))))))) --> a1(b(b(b(b(a1(b(x)))))))
   derived from: 109 and 13797
New Rule (14477): b(b(b(b(a1(b(x)))))))) --> a1(a1(a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(b(a1(x))))))))))))))))
   derived from: 488 and 14093
New Rule (14620): b(b(b(a1(a1(a1(b(a(b(a1(b(a1(b(x)))))))))))))) --> a1(a1(a1(b(b(a(a(a(x)))))))))
   derived from: 2321 and 14477
New Rule (13778): b(a(b(a1(b(b(b(a1(x)))))))))) --> a(b(a1(b(a1(b(a1(a1(a1(b(a(b(a(a(x))))))))))))))
   derived from: 1243 and 13747
New Rule (14998): a(b(a1(b(a1(b(a1(a1(a1(b(a(b(a(a(a(x))))))))))))))) --> b(a(b(a1(b(b(b(x)))))))
   derived from: 19 and 13778
New Rule (15039): b(a1(b(a1(b(a1(a1(a1(b(a(b(a(a(a(x))))))))))))))) --> a1(b(a(b(a1(b(b(b(x)))))))))
   derived from: 19 and 14998
New Rule (5887): b(b(a(a(b(b(a1(b(x)))))))))) --> a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a(x))))))))))))))
   derived from: 488 and 5797
```

```
New Rule (5349): a(b(a1(a1(b(a1(b(b(a(a(b(a1(x))))))))))) --> b(a(b(a1(a1(b(a(b(a1(b(x))))))))))
   derived from: 433 and 3716
New Rule (15431): b(a1(a1(b(a1(b(b(a(a(b(a1(x))))))))))) --> a1(b(a(b(a1(a1(b(a(b(a1(b(x))))))))))
   derived from: 19 and 5349
New Rule (4978): b(a1(a1(b(a1(a1(b(b(b(a(a(x))))))))))) --> a(b(a1(b(a1(b(a(b(a1(a1(b(x))))))))))
   derived from: 380 and 4129
New Rule (4044): a(a(b(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))
                 --> b(a1(a1(b(a1(a1(a1(b(a1(b(a1(x))))))))))
   derived from: 1012 and 0
New Rule (15701): a(b(a1(b(a1(a1(b(a1(b(b(x))))))))) --> b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))))
   derived from: 11398 and 4044
New Rule (15931): b(a1(b(a1(a1(b(a1(b(b(x))))))))) --> a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))))
   derived from: 19 and 15701
New Rule (15977): a(b(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))) --> a1(b(a1(a1(b(a1(a1(a1(b(a(b(x))))))))))
   derived from: 11398 and 15931
New Rule (16226): b(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(b(a(b(x))))))))))))
   derived from: 19 and 15977
New Rule (16367): b(b(a1(a1(a1(b(a1(b(a1(x))))))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(b(a(b(a(x))))))))))))
   derived from: 19 and 16226
New Rule (16541): a1(a1(b(a1(a1(b(a1(a1(a1(b(a(b(a(a(x))))))))))))) --> b(b(a1(a1(a1(b(a1(b(x))))))))
   derived from: 19 and 16367
New Rule (16581): a1(b(a1(a1(b(a1(a1(a1(b(a(b(a(a(x))))))))))))) --> a(b(b(a1(a1(a1(b(a1(b(x))))))))
   derived from: 20 and 16541
New Rule (16646): b(a1(a1(b(a1(a1(a1(b(a(b(a(a(x))))))))))) --> a(a(b(b(a1(a1(a1(b(a1(b(x))))))))))
   derived from: 20 and 16581
New Rule (3588): a(a(b(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))))
                 --> b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))
   derived from: 805 and 1940
New Rule (3501): a(a(b(a(b(a1(a1(a1(b(a(b(b(x))))))))))))
                 --> a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(x))))))))))))))
   derived from: 805 and 3438
New Rule (3465): b(a(a(a(a(b(b(b(a(a(a(b(a(x))))))))))))) --> a(a(a(b(a1(b(a(b(b(x))))))))))
   derived from: 499 and 3438
New Rule (17110): b(a(a(a(a(b(b(b(a(a(a(b(x))))))))))))) --> a(a(a(b(a1(b(a(b(b(a1(x))))))))))
   derived from: 20 and 3465
New Rule (2408): b(b(b(a(a(b(a1(a1(b(x))))))))) --> a1(b(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x))))))))))))
   derived from: 2321 and 2403
New Rule (2037): b(a(b(a1(a1(a1(b(a(b(b(x))))))))))
                 --> a1(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(x))))))))))))))))
   derived from: 657 and 1970
New Rule (1991): b(a1(b(a1(a1(b(a1(a1(b(b(x))))))))))
                 --> a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))))
   derived from: 657 and 1938
New Rule (1029): b(a1(a1(b(a1(a1(b(a1(b(b(a(a(x))))))))))) --> a1(b(a(b(a1(a1(b(b(a1(b(x))))))))))
   derived from: 805 and 1012
New Rule (1002): b(a1(a1(a1(b(a1(a1(b(a(b(b(x))))))))))))
                 --> a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(b(x))))))))))))))
   derived from: 657 and 111
New Rule (939): a1(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a(x))))))))))))))))
                --> a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(b(x))))))))))))))))
   derived from: 488 and 3
New Rule (756): b(a1(b(a(a(b(a1(a1(b(x))))))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(a(x))))))))))))
   derived from: 453 and 739
New Rule (674): a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(b(a(a(x))))))))))))))))
                --> b(a1(b(a1(b(a1(a1(a1(a1(b(x))))))))))
   derived from: 376 and 657
New Rule (524): a1(a1(a1(b(a(b(a1(b(a(b(a1(b(a1(x))))))))))))))
                --> b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(x))))))))))))))
   derived from: 488 and 0
New Rule (438): a(b(a(b(a1(a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(x))))))))))))))))
                --> b(a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(x))))))))))))))
   derived from: 376 and 433
New Rule (384): b(b(a(a(b(a(b(a1(a1(b(x))))))))))) --> a1(b(a1(b(a1(a1(a1(b(a1(b(a(a(x))))))))))))
   derived from: 109 and 376
New Rule (378): a1(b(a1(b(a1(b(a(b(a1(b(a(a(x))))))))))))) --> b(a(b(a1(b(a(b(a1(a1(b(x))))))))))
```

```
   derived from: 358 and 376
New Rule (18972): b(a1(b(a1(b(a(b(a1(b(a(a(x)))))))))) --> a(b(a(b(a1(b(a(b(a1(a1(b(x)))))))))))
   derived from: 20 and 378
New Rule (18880): a1(b(a1(b(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))) --> b(b(a(a(b(a(a(x)))))))
   derived from: 605 and 384
New Rule (19291): b(a1(b(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))) --> a(b(b(a(a(b(a(a(x)))))))
   derived from: 20 and 18880
New Rule (18420): a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))))
                  --> b(a1(b(a(a(a(x))))))
   derived from: 605 and 756
New Rule (19705): a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))))
                  --> a(b(a1(b(a(a(a(x)))))))
   derived from: 20 and 18420
New Rule (19906): b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))) --> a(a(b(a1(b(a(a(a(x)))))))
   derived from: 20 and 19705
New Rule (17853): a1(a1(b(a1(a1(b(a1(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))))))
                  --> b(a(a(b(x))))
   derived from: 953 and 1991
New Rule (20256): a1(b(a1(a1(b(a1(a1(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))))))
                  --> a(b(a(a(b(x)))))
   derived from: 20 and 17853
New Rule (20402): b(a1(a1(b(a1(a1(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))))) --> a(a(b(a(a(b(x))))))
   derived from: 20 and 20256
New Rule (20578): b(a1(a1(b(a1(a1(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))))))) --> a(a(b(a(a(b(a(x)))))))
   derived from: 19 and 20402
New Rule (17482): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))))) --> b(b(b(a(a(a(x))))))
   derived from: 605 and 2408
New Rule (20996): b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(x))))))))))))))))) --> a(b(b(b(a(a(a(x)))))))
   derived from: 20 and 17482
New Rule (17269): b(a(a(a(a(b(a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(x))))))))))))))))))) --> a(a(a(b(a(x)))))
   derived from: 739 and 17110
New Rule (21361): b(a(a(a(a(b(a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(x))))))))))))))))))) --> a(a(a(b(a(a(x))))))
   derived from: 19 and 17269
New Rule (21480): a(a(a(b(a(a(b(a(b(b(a1(x))))))))))) --> b(a(a(a(a(b(b(b(a(x))))))))))
   derived from: 2142 and 21361
New Rule (21629): a(a(b(a(a(b(a(b(b(a1(x))))))))))) --> a1(b(a(a(a(a(b(b(b(a(x))))))))))
   derived from: 19 and 21480
New Rule (21749): a(b(a(a(b(a(b(b(a1(x))))))))) --> a1(a1(b(a(a(a(a(b(b(b(a(x))))))))))
   derived from: 19 and 21629
New Rule (21869): b(a(a(b(a(b(b(a1(x))))))))) --> a1(a1(a1(b(a(a(a(a(b(b(b(a(x))))))))))))
   derived from: 19 and 21749
New Rule (22038): a1(a1(a1(b(a(a(a(a(b(b(b(a(a(x))))))))))))) --> b(a(a(b(a(b(b(x)))))))
   derived from: 19 and 21869
New Rule (22085): a1(a1(b(a(a(a(a(b(b(b(a(a(x))))))))))))) --> a(b(a(a(b(a(b(b(x)))))))
   derived from: 20 and 22038
New Rule (22143): a1(b(a(a(a(a(b(b(b(a(a(x))))))))))) --> a(a(b(a(a(b(a(b(b(x))))))))
   derived from: 20 and 22085
New Rule (22226): b(a(a(a(a(b(b(b(a(a(x))))))))))) --> a(a(a(b(a(a(b(a(b(b(x))))))))))
   derived from: 20 and 22143
New Rule (22300): b(a(b(a(a(b(a(b(b(x))))))))) --> a1(b(a1(a1(b(a1(a1(b(a(a(b(a(a(x))))))))))))))
   derived from: 5180 and 22226
New Rule (22261): b(b(a(a(b(b(a(a(b(a(b(b(x))))))))))))) --> a1(b(b(a1(a1(a1(b(b(a1(b(x))))))))))
   derived from: 14093 and 22226
New Rule (21507): b(a(a(a(a(b(a1(b(a1(b(a1(a1(a1(a1(b(a1(x))))))))))))))))) --> a(a(a(b(a(a(a(x)))))))
   derived from: 19 and 21361
New Rule (22936): b(a(a(a(a(b(a1(b(a1(b(a1(a1(a1(a1(b(x))))))))))))))))) --> a(a(a(b(a(a(a(a(x)))))))
   derived from: 19 and 21507
New Rule (16105): a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))))))))))
                  --> a(b(b(a1(a1(a1(x))))))
   derived from: 739 and 15931
New Rule (23353): b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))))))))))
                  --> a(a(b(b(a1(a1(a1(x)))))))
   derived from: 20 and 16105
New Rule (23513): b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                  --> a(a(b(b(a1(a1(x))))))
```

```
   derived from: 5045 and 23353
New Rule (16080): b(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))))))))))
                  --> a(b(a(b(b(x)))))
   derived from: 1938 and 15931
New Rule (15457): b(a1(b(a1(b(a1(a1(b(a1(a1(b(a1(b(a(b(b(x))))))))))))))))) --> a1(b(a(b(a1(b(a1(x)))))))
   derived from: 10941 and 15431
New Rule (15234): a(b(a1(a1(b(a1(a1(b(a1(b(a(b(b(a1(b(x))))))))))))))) --> b(a(a(a(a(b(b(a(x)))))))))
   derived from: 5311 and 5887
New Rule (24435): b(a1(a1(b(a1(a1(b(a1(b(a(b(b(a1(b(x))))))))))))) --> a1(b(a(a(a(a(b(b(a(x)))))))))
   derived from: 19 and 15234
New Rule (13844): a(b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))))))))
                  --> b(a(b(a(a(b(a1(x)))))))
   derived from: 3659 and 13747
New Rule (12509): a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))))) --> b(a1(x))
   derived from: 5797 and 12140
New Rule (24914): a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))))) --> a(b(a1(x)))
   derived from: 20 and 12509
New Rule (25067): a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))))) --> a(a(b(a1(x))))
   derived from: 20 and 24914
New Rule (25224): a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))) --> a(a(a(b(a1(x)))))
   derived from: 20 and 25067
New Rule (25383): a1(a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))) --> a(a(a(a(b(a1(x))))))
   derived from: 20 and 25224
New Rule (25556): a1(b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))) --> a(a(a(a(a(b(a1(x)))))))
   derived from: 20 and 25383
New Rule (25752): b(a(b(a1(b(a(a(b(a(a(a(b(a(b(b(x))))))))))))))))) --> a(a(a(a(a(a(b(a1(x)))))))
   derived from: 20 and 25556
New Rule (25968): b(a(b(a1(b(a(b(a1(b(a1(a1(b(b(x))))))))))))))) --> a(a(a(a(a(b(a1(b(a1(a1(x))))))))))
   derived from: 336 and 25752
New Rule (26183): b(a(b(a1(b(a(b(a1(b(a1(a1(b(a(x))))))))))))))) --> a(a(a(a(a(a(b(b(a(a(x))))))))))
   derived from: 605 and 25968
New Rule (26333): b(a(b(a1(b(a(b(a1(b(a1(a1(b(x))))))))))))) --> a(a(a(a(a(a(b(b(a(x)))))))))
   derived from: 20 and 26183
New Rule (12075): a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))))))) --> a(x)
   derived from: 5797 and 12012
New Rule (26739): a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))))))) --> a(a(x))
   derived from: 20 and 12075
New Rule (26893): a1(a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))))))) --> a(a(a(x)))
   derived from: 20 and 26739
New Rule (27051): a1(a1(a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))))) --> a(a(a(a(x))))
   derived from: 20 and 26893
New Rule (27211): a1(a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))) --> a(a(a(a(a(x)))))
   derived from: 20 and 27051
New Rule (27386): a1(b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))) --> a(a(a(a(a(a(x))))))
   derived from: 20 and 27211
New Rule (27584): b(a(b(a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))))) --> a(a(a(a(a(a(a(x)))))))
   derived from: 20 and 27386
New Rule (27730): b(a(b(a1(b(a(b(a1(b(a1(a1(a1(b(a(x))))))))))))))) --> a(a(a(a(a(a(b(a(b(x)))))))))
   derived from: 5797 and 27584
New Rule (27946): b(a(b(a1(b(a(b(a1(b(a1(a1(a1(b(x))))))))))))))) --> a(a(a(a(a(a(b(a(b(a1(x))))))))))
   derived from: 20 and 27730
New Rule (27987): b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(a1(b(x))))))))))))))))) --> a(a(a(b(a(b(a1(x)))))))
   derived from: 524 and 27946
New Rule (10382): a1(b(a1(b(a1(b(a1(a1(a1(b(b(a1(b(x))))))))))))))) --> b(a(a(a(b(a(a(b(a(a(x))))))))))
   derived from: 358 and 10239
New Rule (28652): b(a1(b(a1(b(a1(a1(a1(b(b(a1(b(x))))))))))))) --> a(b(a(a(a(b(a(a(b(a(a(x))))))))))
   derived from: 20 and 10382
New Rule (9710): b(a1(b(a1(b(b(a1(a1(a1(a1(b(a1(a1(b(a1(a1(x))))))))))))))))) --> a(a(a(a(b(b(b(x)))))))
   derived from: 336 and 9571
New Rule (29131): b(a1(b(a1(b(b(a1(a1(a1(a1(b(a1(a1(b(a1(x))))))))))))))) --> a(a(a(a(b(b(b(a(x)))))))))
   derived from: 19 and 9710
New Rule (29373): b(a1(b(a1(b(b(a1(a1(a1(a1(b(a1(a1(b(x))))))))))))))) --> a(a(a(a(b(b(b(a(a(x))))))))))
   derived from: 19 and 29131
New Rule (29516): b(a1(b(a1(b(b(a1(a1(a1(a1(b(a(a(b(a1(x))))))))))))))) --> a(a(a(a(b(a1(b(x)))))))
   derived from: 4228 and 29373
```

```
New Rule (29867): b(a1(b(a1(b(a1(b(b(a1(a1(a1(a1(b(a(a(b(x)))))))))))))) --> a(a(a(a(b(a1(b(a(x)))))))
   derived from: 19 and 29516
New Rule (9437): a(b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(a1(x))))))))))))))))))
                 --> b(a(a(b(b(x)))))
   derived from: 0 and 0
New Rule (9250): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(x))))))))))))))))))
                 --> b(a(a(a(b(b(a1(x)))))))
   derived from: 358 and 1243
New Rule (30425): b(b(b(a1(b(a(a(b(b(a1(x))))))))) --> a1(b(b(b(a1(a1(a1(a1(a1(b(a(b(x))))))))))))
   derived from: 380 and 9250
New Rule (30670): b(b(b(a1(b(a(a(b(b(x)))))))) --> a1(b(b(b(a1(a1(a1(a1(a1(b(a(b(a(x))))))))))))
   derived from: 19 and 30425
New Rule (9028): a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(a(x))))))))))))))))))) --> b(b(x))
   derived from: 8270 and 1297
New Rule (31006): a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                  --> b(b(a1(x)))
   derived from: 20 and 9028
New Rule (31196): b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                  --> a1(b(b(a1(x))))
   derived from: 19 and 31006
New Rule (31455): a1(b(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                  --> b(b(b(a1(x))))
   derived from: 358 and 31196
New Rule (31647): b(a1(b(a1(a1(b(b(b(b(a1(x))))))))) --> a1(a1(b(a(a(a(a(a(a(a(b(x))))))))))
   derived from: 1938 and 31455
New Rule (31915): b(a1(b(a1(a1(b(b(b(b(x))))))))) --> a1(a1(b(a(a(a(a(a(a(a(b(a(x))))))))))
   derived from: 19 and 31647
New Rule (32167): a1(b(a(b(a1(b(a1(a1(b(a(a(a(a(b(a(x))))))))))))))) --> a(a(b(b(b(b(b(x)))))))
   derived from: 805 and 31915
New Rule (32302): b(a(b(a1(b(a1(a1(b(a(a(a(a(b(a(x))))))))))))))) --> a(a(a(b(b(b(b(b(x))))))))
   derived from: 20 and 32167
New Rule (32448): a(a(a(b(b(b(b(b(a1(x))))))))) --> b(a(b(a1(b(a1(a1(b(a(a(a(a(b(x))))))))))))
   derived from: 20 and 32302
New Rule (32593): a(a(b(b(b(b(b(a1(x))))))))) --> a1(b(a(b(a1(b(a1(a1(b(a(a(a(a(b(x))))))))))))))
   derived from: 19 and 32448
New Rule (32739): a(b(b(b(b(b(a1(x)))))))) --> a1(a1(b(a(b(a1(b(a1(a1(b(a(a(a(a(b(x))))))))))))))
   derived from: 19 and 32593
New Rule (32854): a(b(b(a(a(b(a1(a1(a1(b(a(a(a(a(b(x))))))))))))))) --> b(b(b(b(a1(x)))))
   derived from: 5045 and 32739
New Rule (33095): b(b(a(a(b(a1(a1(a1(b(a(a(a(a(b(x))))))))))))))) --> a1(b(b(b(b(a1(x))))))
   derived from: 19 and 32854
New Rule (33344): b(b(b(b(b(a1(x)))))))) --> a1(a1(a1(b(a(b(a1(b(a1(a1(b(a(a(a(a(b(x))))))))))))))))
   derived from: 488 and 33095
New Rule (32141): b(a1(a1(b(a(b(a1(a1(b(a(a(a(a(a(a(b(a(x))))))))))))))))) --> b(b(b(b(x))))
   derived from: 2142 and 31915
New Rule (33718): a1(b(a(b(a1(a1(b(a(a(a(a(a(a(a(b(a(x))))))))))))))))) --> a(b(b(b(x))))
   derived from: 605 and 32141
New Rule (33837): b(a(b(a1(a1(b(a(a(a(a(a(a(b(a(x))))))))))))))) --> a(a(b(b(b(x)))))
   derived from: 20 and 33718
New Rule (33981): b(a(b(a1(a1(b(a(a(a(a(a(a(b(x))))))))))))))) --> a(a(b(b(b(a1(x))))))
   derived from: 20 and 33837
New Rule (31684): b(a1(b(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))) --> a(b(b(b(a1(x)))))
   derived from: 20 and 31455
New Rule (8382): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a(b(a1(b(a(a(x))))))))))))))))))
                 --> b(b(a(a(a(a(b(a(x))))))))
   derived from: 358 and 8270
New Rule (6962): a(b(a1(a1(b(a1(b(a1(a1(a1(b(a(b(a1(b(x))))))))))))))) --> b(a(a(a(a(a(b(a1(x))))))))
   derived from: 0 and 6086
New Rule (34822): b(a1(a1(b(a1(b(a1(a1(a1(b(a(b(a1(b(x))))))))))))))) --> a1(b(a(a(a(a(a(b(a1(x))))))))
   derived from: 19 and 6962
New Rule (5334): b(a1(a1(b(a1(a1(a1(b(a1(a1(b(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))))))))
                 --> a1(b(a(b(a(a(b(a1(x))))))))
   derived from: 1012 and 3716
New Rule (5005): b(a1(a1(b(a1(a1(b(a(a(b(a(b(a1(a1(b(x))))))))))))))) --> a(b(a(b(a(a(a(a(a(x))))))))
   derived from: 376 and 4129
```

```
New Rule (3536): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                 --> b(a1(b(a1(x))))
   derived from: 3107 and 1940
New Rule (35817): b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x))))))))))))))))))
                  --> a(b(a1(b(a1(x)))))
   derived from: 20 and 3536
New Rule (3529): b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(a(x))))))))))))))))))))
                 --> a(b(b(a(a(a(a(b(a(x))))))))))
   derived from: 3438 and 1940
New Rule (3508): b(a1(b(a1(a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(b(x))))))))))))))))))
                 --> a(b(a(a(a(b(b(a1(x)))))))))
   derived from: 657 and 3438
New Rule (3356): b(a1(b(a1(a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a1(b(x))))))))))))))))))
                 --> a(b(b(a(a(a(a(b(b(a(x))))))))))
   derived from: 488 and 3264
New Rule (2973): a(b(a1(a1(b(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(a1(b(x))))))))))))))))))))
                 --> b(a(b(a(a(a(a(b(a(a(x))))))))))
   derived from: 433 and 2907
New Rule (2438): a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))))) --> b(b(a(a(a(b(a(a(x))))))))
   derived from: 862 and 2403
New Rule (37323): b(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))))) --> a(b(b(a(a(a(b(a(a(x))))))))))
   derived from: 20 and 2438
New Rule (1417): b(a1(a1(b(a1(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a(b(a1(b(a(x))))))))))))))))))))
                 --> a(b(a1(b(a1(a1(x))))))
   derived from: 890 and 12
New Rule (1311): a1(b(a1(a1(b(a1(a1(b(a1(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))))))))
                 --> a(b(a(a(a(b(a(b(a(a(x))))))))))
   derived from: 376 and 862
New Rule (1255): a(b(a1(a1(b(a1(b(b(a1(a1(a1(a1(b(a(b(b(x))))))))))))))))) --> b(a(a(b(b(a(x))))))
   derived from: 433 and 1219
New Rule (38145): b(a1(a1(b(a1(b(b(a1(a1(a1(a1(b(a(b(b(x))))))))))))))) --> a1(b(a(a(b(b(a(x)))))))
   derived from: 19 and 1255
New Rule (829): b(a1(a1(b(a1(a1(a1(a1(a1(a1(b(a(b(a1(b(a(b(a(x))))))))))))))))))
                --> a(a(b(a(a(a(b(b(a(x))))))))))
   derived from: 488 and 805
New Rule (818): a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))))))) --> b(b(b(a(a(x)))))
   derived from: 739 and 805
New Rule (38832): a1(b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))))))) --> a(b(b(b(a(a(x))))))
   derived from: 20 and 818
New Rule (39099): b(a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(b(a1(b(x))))))))))))))) --> a(a(b(b(b(a(a(x)))))))
   derived from: 20 and 38832
New Rule (37735): b(a(b(a(a(a(a(b(a1(b(a1(a1(x))))))))))))) --> a1(a1(a1(b(a1(a1(b(a(a(a(b(x))))))))))
   derived from: 12963 and 1417
New Rule (39624): b(a(b(a(a(a(a(b(a1(b(a1(x))))))))))) --> a1(a1(a1(b(a1(a1(b(a(a(a(b(a(x))))))))))
   derived from: 19 and 37735
New Rule (39890): b(a(b(a(a(a(a(b(a1(b(x))))))))))) --> a1(a1(a1(b(a1(a1(b(a(a(a(a(b(a(a(x))))))))))))
   derived from: 19 and 39624
New Rule (40192): b(a1(a1(a1(b(a1(a1(b(a(a(a(b(a(a(x))))))))))))))) --> a1(b(a1(b(a(a(a(a(b(a1(b(x))))))))))
   derived from: 109 and 39890
New Rule (40328): a1(b(a1(b(a(a(a(b(a1(b(a1(x))))))))))))
                  --> b(a1(a1(a1(b(a1(a1(b(a(a(a(b(a(x))))))))))))))
   derived from: 20 and 40192
New Rule (40554): b(a1(b(a(a(a(b(a1(b(a1(x))))))))))) --> a(b(a1(a1(a1(b(a1(a1(b(a(a(a(b(a(x))))))))))))))
   derived from: 20 and 40328
New Rule (40189): b(a1(a1(b(a1(a1(b(a(a(a(b(a(a(x))))))))))))))) --> a(b(a1(a1(b(a(a(a(a(b(a1(b(x))))))))))))
   derived from: 336 and 39890
New Rule (40993): a(b(a1(a1(b(a(a(a(b(a1(b(a1(x))))))))))))) --> b(a1(a1(b(a1(a1(b(a(a(a(b(a(x))))))))))))
   derived from: 20 and 40189
New Rule (41191): b(a1(a1(b(a(a(a(b(a1(b(a1(x))))))))))))
                  --> a1(b(a1(a1(b(a1(a1(b(a(a(a(b(a(x))))))))))))))
   derived from: 19 and 40993
New Rule (40920): a1(b(a(b(a1(b(a1(a1(b(b(a(a(x))))))))))))) --> b(a1(b(a1(a1(b(a(a(a(a(b(a1(b(x))))))))))))
   derived from: 13940 and 40189
New Rule (41579): b(a(b(a1(b(a1(a1(b(b(a(a(x))))))))))) --> a(b(a1(b(a1(a1(b(a(a(a(b(a1(b(x))))))))))))
   derived from: 20 and 40920
```

```
New Rule (41679): a(b(a(a(a(a(b(a1(a1(a1(b(b(a(a(x)))))))))))))) --> b(b(a(a(a(a(b(a1(b(x)))))))))
   derived from: 6086 and 41579
New Rule (41804): b(b(a(a(a(a(a(b(a1(b(a1(x)))))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(b(b(a(x)))))))))))))
   derived from: 20 and 41679
New Rule (42039): b(b(a(a(a(a(a(a(b(b(a(x)))))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(b(x)))))))))))
   derived from: 3107 and 41804
New Rule (42265): b(b(a(a(a(a(a(a(b(b(x)))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(b(a1(x)))))))))))
   derived from: 20 and 42039
New Rule (42547): b(b(a(a(a(b(a(b(b(x)))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(b(a1(a1(b(a1(x)))))))))))))))
   derived from: 657 and 42265
New Rule (42024): b(b(a(a(a(a(a(a(b(a(b(a1(x)))))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(a1(b(x)))))))))))
   derived from: 1940 and 41804
New Rule (43128): b(b(a(a(a(a(a(a(b(a(b(x)))))))))))) --> a(b(a(a(a(a(b(a1(a1(a1(a1(b(a(x)))))))))))))
   derived from: 19 and 42024
New Rule (43386): a(b(a(a(a(b(a(b(a(a(a(a(a(b(a(b(x)))))))))))))))) --> a1(b(a1(a1(a1(b(a(x)))))))
   derived from: 3438 and 43128
New Rule (43677): b(a(a(a(b(a(b(a(a(a(a(a(b(a(b(x)))))))))))))))) --> a1(a1(b(a1(a1(a1(b(a(x)))))))))
   derived from: 19 and 43386
New Rule (43974): a(a(a(a(b(b(b(b(b(x)))))))))) --> b(a1(b(a1(a1(a1(b(a1(a1(a1(b(a(x)))))))))))))
   derived from: 376 and 43677
New Rule (44213): a(a(a(b(b(b(b(b(b(x)))))))))) --> a1(b(a1(b(a1(a1(a1(b(a1(a1(a1(b(a(x)))))))))))))
   derived from: 19 and 43974
New Rule (44445): a(a(b(b(b(b(b(b(x)))))))) --> a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(a1(b(a(x)))))))))))))))
   derived from: 19 and 44213
New Rule (44678): a(b(b(b(b(b(b(x)))))))) --> a1(a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(a1(b(a(x)))))))))))))))
   derived from: 19 and 44445
New Rule (44925): b(b(b(b(b(b(x)))))) --> a1(a1(a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(a1(b(a(x)))))))))))))))))
   derived from: 19 and 44678
New Rule (43979): a(b(a1(b(a1(a1(b(a(a(a(a(b(a(b(x)))))))))))))) --> b(a(b(a1(b(a1(a1(a1(b(a(x)))))))))))
   derived from: 336 and 43677
New Rule (45479): b(a1(b(a1(a1(b(a(a(a(a(b(a(b(x)))))))))))))) --> a1(b(a(b(a1(b(a1(a1(a1(b(a(x)))))))))))
   derived from: 19 and 43979
New Rule (41803): b(a(a(a(a(b(a1(a1(a1(b(b(a(a(x)))))))))))))) --> a1(b(b(a(a(a(a(a(b(a1(b(x)))))))))))
   derived from: 19 and 41679
New Rule (36502): a(b(a1(a1(a1(b(a(b(a1(b(a1(b(a1(a1(x)))))))))))))))
                  --> b(a1(b(a1(b(a1(a1(b(a1(b(x)))))))))
   derived from: 2403 and 3508
New Rule (46128): b(a1(a1(a1(b(a(b(a1(b(a1(b(a1(a1(x)))))))))))))
                  --> a1(b(a1(b(a1(b(a1(a1(b(a1(b(x)))))))))))
   derived from: 19 and 36502
New Rule (46241): b(a1(b(a1(b(a1(a1(a1(a1(b(a1(a1(a1(a1(b(a1(b(x)))))))))))))))))
                  --> a1(a1(a1(b(b(a(x))))))
   derived from: 14620 and 46128
New Rule (46356): b(a1(a1(a1(b(a(b(a1(b(a1(b(a1(x)))))))))))))
                  --> a1(b(a1(b(a1(b(a1(a1(b(a1(b(a(x)))))))))))))
   derived from: 19 and 46128
New Rule (46990): a1(a1(b(a(b(a1(b(a1(b(a1(x)))))))))))
                  --> a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(x)))))))))))))))
   derived from: 605 and 46356
New Rule (47218): a1(b(a(b(a1(b(a1(b(a1(x)))))))))
                  --> a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(x)))))))))))))))
   derived from: 20 and 46990
New Rule (47467): b(a(b(a1(b(a1(b(a1(x)))))))))
                  --> a(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(x)))))))))))))))))
   derived from: 20 and 47218
New Rule (47756): a(a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x)))))))))))))))))
                  --> b(a(b(a1(b(a1(b(x)))))))
   derived from: 19 and 47467
New Rule (47836): a(a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x)))))))))))))))))
                  --> a1(b(a(b(a1(b(a1(b(x)))))))))
   derived from: 19 and 47756
New Rule (47917): a(a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x)))))))))))))))))
                  --> a1(a1(b(a(b(a1(b(a1(b(x)))))))))
   derived from: 19 and 47836
New Rule (48003): a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x)))))))))))))))
```

```
                --> a1(a1(a1(b(a(b(a1(b(a1(b(x)))))))))
   derived from: 19 and 47917
New Rule (48088): b(a1(a1(b(a1(a1(a1(a1(b(a1(b(a(a(x)))))))))))))
                --> a1(a1(a1(a1(b(a(b(a1(b(a1(b(x)))))))))))
   derived from: 19 and 48003
New Rule (36060): b(a1(b(a1(a1(b(a1(a1(a1(b(a1(x)))))))))))
                --> a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(x)))))))))))))
   derived from: 2142 and 35817
New Rule (48537): a1(a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(a(x)))))))))))))
                --> b(a1(b(a1(a1(b(a1(a1(a1(b(x)))))))))))
   derived from: 19 and 36060
New Rule (48647): a1(b(a1(a1(a1(a1(b(a1(a1(b(a(b(a(x)))))))))))))
                --> a(b(a1(b(a1(a1(b(a1(a1(a1(b(x)))))))))))
   derived from: 20 and 48537
New Rule (48789): b(a1(a1(a1(a1(b(a1(a1(b(a(b(a(x)))))))))))))
                --> a(a(b(a1(b(a1(a1(b(a1(a1(a1(b(x)))))))))))))
   derived from: 20 and 48647
New Rule (35926): a(a(b(a(b(a1(a1(b(b(b(a1(b(x)))))))))))))
                --> b(a1(b(a1(a1(a1(a1(b(a1(b(b(a(x)))))))))))))
   derived from: 5887 and 35817
New Rule (49057): b(a1(b(a1(a1(a1(a1(b(a1(b(a(b(a(a(a(b(b(a(x)))))))))))))))))) --> a(a(b(a(b(x)))))
   derived from: 24435 and 35926
New Rule (49441): b(a1(b(a1(a1(a1(a1(b(a1(b(a(b(a(a(a(b(b(x)))))))))))))))))) --> a(a(b(a(b(a1(x))))))
   derived from: 20 and 49057
New Rule (49729): b(a1(a1(b(a(b(a(a(b(a(b(a1(x)))))))))))) --> a1(a1(b(a1(b(a(b(a(a(a(a(b(b(x)))))))))))))
   derived from: 2142 and 49441
New Rule (50055): a1(a1(b(a1(b(a(b(a(a(a(b(b(a(x)))))))))))))) --> b(a1(a1(b(a(b(a(a(a(b(a(b(x)))))))))))
   derived from: 19 and 49729
New Rule (50165): a1(b(a1(b(a(b(a(a(a(b(b(a(x)))))))))))) --> a(b(a1(a1(b(a(b(a(a(a(b(a(b(x)))))))))))))
   derived from: 20 and 50055
New Rule (50309): b(a1(b(a(b(a(a(a(b(b(a(x)))))))))))) --> a(a(b(a1(a1(b(a(b(a(a(a(b(a(b(x)))))))))))))
   derived from: 20 and 50165
New Rule (49231): a(b(a(b(a1(a1(b(b(b(a1(b(x)))))))))))
                --> a1(b(a1(b(a1(a1(a1(a1(b(a1(b(b(a(x)))))))))))))
   derived from: 19 and 35926
New Rule (50789): b(a(b(a1(a1(b(b(b(a1(b(x)))))))))))
                --> a1(a1(b(a1(b(a1(a1(a1(a1(b(a1(b(b(a(x)))))))))))))))
   derived from: 19 and 49231
New Rule (35549): a1(b(a(b(a(a(b(a(b(a(a1(b(x)))))))))))))
                --> b(a1(a1(b(a1(a1(a1(b(a1(b(a(a(x)))))))))))))
   derived from: 1012 and 5005
New Rule (51439): b(a(b(a(a(b(a(b(a(a1(b(x)))))))))))) --> a(b(a1(a1(b(a1(a1(a1(b(a1(b(a(a(x)))))))))))))
   derived from: 20 and 35549
New Rule (35544): b(a1(a1(b(a1(a1(b(a(a(b(a(a(a(b(a(x)))))))))))))))) --> a(b(a(b(a1(b(a(b(b(x)))))))))
   derived from: 1219 and 5005
New Rule (51983): a(b(a(b(a1(b(a(b(b(a1(x)))))))))) --> b(a1(a1(b(a1(a1(b(a(a(b(a(a(a(b(x)))))))))))))))
   derived from: 20 and 35544
New Rule (52166): b(a(b(a1(b(a(b(b(a1(x)))))))))) --> a1(b(a1(a1(b(a1(a1(b(a(a(b(a(a(a(b(x)))))))))))))))
   derived from: 19 and 51983
New Rule (34693): b(a1(b(a1(a1(b(a(b(b(x)))))))))
                --> a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(b(a(x)))))))))))))))
   derived from: 13778 and 0
New Rule (52758): b(a1(b(a(b(b(a1(b(x)))))))))
                --> a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(b(a(a(x)))))))))))))))
   derived from: 488 and 34693
New Rule (53071): a(b(a1(a1(a1(b(a1(a1(b(a(b(x)))))))))))
                --> a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(x)))))))))))))
   derived from: 3107 and 52758
New Rule (53386): b(a1(a1(a1(b(a1(a1(b(a(b(x)))))))))))
                --> a1(a1(a1(b(a1(a1(b(a1(a1(a1(a1(b(a(x)))))))))))))
   derived from: 19 and 53071
New Rule (52525): a(b(b(b(a1(b(a(a(b(a1(b(a(x)))))))))))) --> b(b(b(a1(a1(a1(a1(b(a(a(b(x)))))))))))
   derived from: 29867 and 34693
New Rule (53840): a(b(b(b(a1(b(a(a(b(a1(b(x)))))))))))) --> b(b(b(a1(a1(a1(a1(b(a(a(b(a1(x)))))))))))))
   derived from: 20 and 52525
```

```
New Rule (54127): b(b(b(a1(b(a(a(b(a1(b(x))))))))))
                  --> a1(b(b(b(a1(a1(a1(a1(a1(b(a(a(b(a1(x))))))))))))))
   derived from: 19 and 53840
New Rule (34216): b(a(a(b(b(b(a1(x)))))))) --> a1(b(a1(b(a1(a1(a1(a1(a1(b(a(a(a(a(a(a(a(b(x))))))))))))))))))
   derived from: 109 and 33981
New Rule (54731): a1(b(a1(b(a1(a1(a1(a1(b(a(a(a(a(a(a(a(b(a(x))))))))))))))))))) --> b(a(a(b(b(b(x))))))
   derived from: 19 and 34216
New Rule (54873): b(a1(b(a1(a1(a1(a1(b(a(a(a(a(a(a(a(b(a(x))))))))))))))))) --> a(b(a(a(b(b(b(x)))))))
   derived from: 20 and 54731
New Rule (33475): a1(b(a(a(b(a1(b(b(b(a1(x))))))))))) --> b(a1(a1(b(b(a1(a1(a1(b(a(a(a(a(b(x))))))))))))))
   derived from: 4228 and 33344
New Rule (55290): b(a(a(b(a1(b(b(b(a1(x))))))))) --> a(b(a1(a1(b(b(a1(a1(a1(b(a(a(a(a(b(x))))))))))))))))
   derived from: 20 and 33475
New Rule (55549): a(b(a1(a1(b(b(a1(a1(a1(b(a(a(a(a(b(a(x))))))))))))))))) --> b(a(a(b(a1(b(b(b(x)))))))))
   derived from: 19 and 55290
New Rule (55663): b(a1(a1(b(b(a1(a1(a1(b(a(a(a(a(b(a(x))))))))))))))))) --> a1(b(a(a(b(a1(b(b(b(x)))))))))
   derived from: 19 and 55549
New Rule (33352): b(a1(a1(a1(b(a(b(a1(b(a1(a1(b(a(a(a(a(b(x))))))))))))))))))
                  --> a1(a1(a1(a1(b(a1(b(a1(a1(a1(b(a1(a1(a1(b(x))))))))))))))))
   derived from: 109 and 33095
New Rule (31292): a1(b(b(a1(b(a(a(b(b(a1(b(x))))))))))))
                  --> b(a1(a1(b(a1(a1(a1(a1(b(a1(b(b(a(x))))))))))))))
   derived from: 5887 and 31196
New Rule (56475): b(b(a1(b(a(a(b(b(a1(b(x))))))))))
                  --> a(b(a1(a1(b(a1(a1(a1(a1(b(a1(b(b(a(x))))))))))))))
   derived from: 20 and 31292
New Rule (27738): a1(b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))))) --> b(a1(b(a1(a1(b(a(a(a(a(a(x))))))))))))
   derived from: 5045 and 27584
New Rule (57130): b(a(a(b(a(a(a(a(a(a(b(b(x))))))))))))) --> a(b(a1(b(a1(a1(b(a(a(a(a(a(x))))))))))))
   derived from: 20 and 27738
New Rule (57439): b(a(a(b(a(a(a(b(a(b(b(x))))))))))) --> a(b(a1(b(a1(a1(b(a(a(a(a(b(a1(x))))))))))))))
   derived from: 657 and 57130
New Rule (25646): a(b(a(a(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))))))
                  --> a1(b(a1(a1(b(a1(a1(b(b(a1(a1(x))))))))))))
   derived from: 9571 and 0
New Rule (58085): b(a(a(b(a1(b(a1(a1(b(a1(a1(b(x))))))))))))
                  --> a1(a1(b(a1(a1(b(a1(a1(b(b(a1(a1(x))))))))))))
   derived from: 19 and 25646
New Rule (25349): b(a(b(b(a1(b(a(a(b(a1(x))))))))))) --> a(a(a(b(a(b(a1(a1(a1(b(a(b(a1(b(x))))))))))))))
   derived from: 2142 and 0
New Rule (23996): a1(b(a(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))
                  --> b(a1(a1(b(a1(a1(b(a1(b(b(x))))))))))
   derived from: 111 and 16080
New Rule (58958): b(a(b(a1(a1(a1(a1(a1(b(a(b(b(a1(x))))))))))))))
                  --> a(b(a1(a1(b(a1(a1(b(a1(b(b(x))))))))))))
   derived from: 20 and 23996
New Rule (59216): a(b(a1(a1(b(a1(a1(b(a1(b(b(a(x))))))))))))
                  --> b(a(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))
   derived from: 19 and 58958
New Rule (59348): b(a1(a1(b(a1(a1(b(a1(b(b(a(x))))))))))))
                  --> a1(b(a(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))))
   derived from: 19 and 59216
New Rule (23339): a(b(b(a1(a1(a1(a1(a1(b(a(b(b(x))))))))))))
                  --> a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(x))))))))))))
   derived from: 605 and 0
New Rule (59847): b(b(a1(a1(a1(a1(b(a(b(b(x))))))))))
                  --> a1(a1(b(a1(a1(b(a1(a1(a1(b(a1(b(a1(x))))))))))))))
   derived from: 19 and 23339
New Rule (23260): a(b(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(x))))))))))))))
                  --> b(a1(b(a1(a1(b(a1(b(a(b(x))))))))))
   derived from: 5852 and 0
New Rule (60334): b(a1(b(a1(a1(b(a1(b(a(b(a1(x))))))))))))
                  --> a(b(b(a1(a1(a1(a1(a1(b(a(b(a1(b(x))))))))))))))
   derived from: 20 and 23260
New Rule (60333): b(b(a1(a1(a1(a1(a1(b(a(b(a1(b(a(x))))))))))))))
```

```
                --> a1(b(a1(b(a1(a1(b(a1(b(a(b(x))))))))))
   derived from: 19 and 23260
New Rule (22946): b(a(a(a(a(b(b(b(b(a(a(a(x)))))))))))) --> a(a(a(b(a(a(a(b(a1(b(a1(a1(b(x))))))))))))
   derived from: 20996 and 22936
New Rule (61009): b(a(a(a(a(b(b(b(b(a(a(x))))))))))) --> a(a(a(b(a(a(a(b(a1(b(a1(a1(b(a1(x))))))))))))))
   derived from: 20 and 22946
New Rule (61185): b(a(a(a(a(b(b(b(b(a(x)))))))))) --> a(a(a(b(a(a(a(b(a1(b(a1(a1(b(a1(a1(x))))))))))))))
   derived from: 20 and 61009
New Rule (61388): b(a(a(a(a(b(b(b(b(x))))))))) --> a(a(a(b(a(a(a(b(a1(b(a1(a1(b(a1(a1(a1(x))))))))))))))
   derived from: 20 and 61185
New Rule (22713): b(a1(b(b(b(a(a(b(a(b(b(x))))))))))) --> a(b(a(b(a1(b(a1(a1(a1(b(b(a1(b(x))))))))))))
   derived from: 3107 and 22261
New Rule (62079): b(a1(b(b(b(a(a(b(a(b(a(x))))))))))) --> a(b(a(b(a1(b(a1(a1(a1(b(b(x))))))))))
   derived from: 605 and 22713
New Rule (62313): b(a1(b(b(b(a(a(b(a(b(x))))))))))) --> a(b(a(b(a1(b(a1(a1(a1(b(b(a1(x))))))))))))
   derived from: 20 and 62079
New Rule (62504): b(a(b(a(a(b(b(a(a(b(a(b(x))))))))))))) --> a(b(a1(b(a1(a1(a1(b(b(a1(x))))))))))
   derived from: 1991 and 62313
New Rule (63031): a(b(a1(a1(b(b(b(a(a(b(a(b(x))))))))))))) --> b(a(a(b(a1(b(a1(a1(a1(b(b(a1(x))))))))))))
   derived from: 336 and 62504
New Rule (63317): b(a1(a1(b(b(b(a(a(b(a(b(x))))))))))) --> a1(b(a(a(b(a1(b(a1(a1(a1(b(b(a1(x))))))))))))))
   derived from: 19 and 63031
New Rule (62975): a1(a1(b(a(b(a1(b(a(b(a(a(b(a(b(x))))))))))))))) --> a(b(b(b(a1(a1(a1(b(b(a1(x))))))))))
   derived from: 1940 and 62504
New Rule (63875): a(b(b(b(a1(a1(a1(b(b(a(a(a(b(a1(a1(b(x))))))))))))))))) --> a(a(a(b(a1(b(a(x)))))))
   derived from: 12963 and 62975
New Rule (64286): b(b(b(a1(a1(a1(b(b(a(a(a(b(a1(a1(b(x))))))))))))))) --> a(a(b(a1(b(a(x))))))
   derived from: 19 and 63875
New Rule (64652): b(b(b(a1(a1(a1(b(b(a(a(a(a(x))))))))))))) --> a(a(b(a1(b(a1(b(a(b(b(x))))))))))
   derived from: 605 and 64286
New Rule (64834): a(a(b(a1(b(a1(b(a(b(b(a1(x))))))))))) --> b(b(b(a1(a1(a1(b(b(a(a(a(x))))))))))
   derived from: 20 and 64652
New Rule (65033): a(b(a1(b(a1(b(a(b(b(a1(x))))))))))) --> a1(b(b(b(a1(a1(a1(b(b(a(a(a(x))))))))))))
   derived from: 19 and 64834
New Rule (65241): b(a1(b(a1(b(a(b(b(a1(x))))))))) --> a1(a1(b(b(b(a1(a1(a1(b(b(a(a(a(x))))))))))))))
   derived from: 19 and 65033
New Rule (64820): b(a(a(a(b(a1(b(a1(a1(a1(b(a1(b(a(b(b(x))))))))))))))))) --> a1(b(b(a(a(a(a(x))))))))
   derived from: 499 and 64652
New Rule (64427): b(b(b(a1(a1(a1(b(b(a(a(a(b(a1(a1(a1(b(b(a1(x))))))))))))))))))) --> a1(b(a(b(b(x)))))
   derived from: 31196 and 64286
New Rule (66166): b(b(b(a1(a1(a1(b(b(a(a(a(b(a1(a1(a1(b(a(x))))))))))))))))))) --> a(a(b(b(x))))
   derived from: 605 and 64427
New Rule (66300): b(b(a(a(a(b(a(b(a(a(b(a(x))))))))))))) --> a(b(a1(a1(b(a1(a1(b(x))))))))
   derived from: 13778 and 66166
New Rule (66589): a(b(a(a(a(b(a(b(a(a(b(a(x))))))))))))) --> a(a(b(a(b(b(x))))))
   derived from: 605 and 66300
New Rule (66738): b(a(a(a(b(a(b(a(a(b(a(x))))))))))) --> a(b(a(b(b(x)))))
   derived from: 19 and 66589
New Rule (66946): b(a(a(a(b(a(b(a(a(b(x))))))))))) --> a(b(a(b(b(a1(x))))))
   derived from: 20 and 66738
New Rule (67131): a(a(a(a(a(b(a1(b(a1(a1(x))))))))))) --> a(b(b(a(a(x)))))
   derived from: 18972 and 66946
New Rule (67454): a(a(a(a(a(b(a1(b(a1(a1(x))))))))))) --> b(b(a(a(x))))
   derived from: 19 and 67131
New Rule (67620): a(a(a(a(b(a1(b(a1(a1(x))))))))) --> a1(b(b(a(a(x)))))
   derived from: 19 and 67454
New Rule (67788): a(a(a(b(a1(b(a1(a1(x))))))))) --> a1(a1(b(b(a(a(x))))))
   derived from: 19 and 67620
New Rule (67957): a(a(b(a1(b(a1(a1(x))))))) --> a1(a1(a1(b(b(a(a(x)))))))
   derived from: 19 and 67788
New Rule (68130): a(b(a1(b(a1(a1(x)))))) --> a1(a1(a1(a1(b(b(a(a(x))))))))
   derived from: 19 and 67957
New Rule (68307): b(a1(b(a1(a1(x))))) --> a1(a1(a1(a1(a1(b(b(a(a(x))))))))))
   derived from: 19 and 68130
New Rule (68499): b(a1(b(a1(x)))) --> a1(a1(a1(a1(a1(b(b(a(a(a(x))))))))))
```

```
   derived from: 19 and 68307
New Rule (68665): a1(a1(a1(a1(a1(b(b(a(a(a(a(x)))))))))) --> b(a1(b(x)))
   derived from: 19 and 68499
New Rule (68704): a1(a1(a1(a1(b(b(a(a(a(a(x)))))))))) --> a(b(a1(b(x))))
   derived from: 20 and 68665
New Rule (68743): a1(a1(a1(b(b(a(a(a(a(x)))))))) --> a(a(b(a1(b(x)))))
   derived from: 20 and 68704
New Rule (68784): a1(a1(b(b(a(a(a(a(x)))))))) --> a(a(a(b(a1(b(x))))))
   derived from: 20 and 68743
New Rule (68835): a1(b(b(a(a(a(a(x))))))) --> a(a(a(a(b(a1(b(x)))))))
   derived from: 20 and 68784
New Rule (68902): b(b(a(a(a(a(x)))))) --> a(a(a(a(a(b(a1(b(x))))))))
   derived from: 20 and 68835
New Rule (67758): a1(b(b(a(a(b(a1(a1(b(a(b(a1(x))))))))))) --> a(a(a(x)))
   derived from: 4129 and 0
New Rule (69145): b(b(a(a(b(a1(a1(b(a(b(a1(x))))))))))) --> a(a(a(a(x))))
   derived from: 20 and 67758
New Rule (69323): b(b(a(a(b(a1(a1(b(a(b(x))))))))))) --> a(a(a(a(a(x)))))
   derived from: 19 and 69145
New Rule (657): b(a1(a1(a1(a1(a1(b(b(a(a(a(x))))))))))) --> a1(a1(a1(b(a(b(b(x)))))))
New Rule (69669): a1(a1(a1(b(a(b(b(a1(x))))))))) --> b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))
   derived from: 20 and 657
New Rule (69768): a1(a1(b(a(b(b(a1(x)))))))) --> a(b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))
   derived from: 20 and 69669
New Rule (69876): b(a1(a1(b(a(b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))))))) --> x
   derived from: 605 and 69768
New Rule (70007): b(a1(a1(b(a(b(a1(a1(a1(a1(a1(b(b(a(x))))))))))))))) --> a1(x)
   derived from: 20 and 69876
New Rule (70154): b(a1(a1(b(a(b(a1(a1(a1(a1(a1(b(b(x))))))))))))) --> a1(a1(x))
   derived from: 20 and 70007
New Rule (70164): b(a(b(a1(a1(a1(a1(a1(b(b(a(a(b(x))))))))))))) --> a(a(x))
   derived from: 68902 and 70154
New Rule (69881): a1(b(a(b(b(a1(x)))))) --> a(a(b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))))
   derived from: 20 and 69768
New Rule (70739): b(a(b(b(a1(x))))) --> a(a(a(b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))))))
   derived from: 20 and 69881
New Rule (70896): b(a1(a1(b(a1(a1(b(a1(a1(a1(a1(a1(b(b(a(a(x))))))))))))))))) --> a(a(b(x)))
   derived from: 1219 and 70739
New Rule (71034): b(a1(a1(b(a1(a1(b(a1(a1(a1(a1(a1(b(b(a(x))))))))))))))))) --> a(a(b(a1(x))))
   derived from: 20 and 70896
New Rule (71175): b(a1(a1(b(a1(a1(b(a1(a1(a1(a1(a1(b(b(x))))))))))))))) --> a(a(b(a1(a1(x)))))
   derived from: 20 and 71034
New Rule (71337): b(b(a(a(b(a(a(x)))))))) --> a(a(b(a1(a1(a1(b(x)))))))
   derived from: 10239 and 71175
New Rule (71526): b(b(a(a(b(a(x)))))) --> a(a(b(a1(a1(a1(b(a1(x))))))))
   derived from: 20 and 71337
New Rule (71665): b(b(a(a(b(x))))) --> a(a(b(a1(a1(a1(b(a1(a1(x))))))))
   derived from: 20 and 71526
New Rule (71673): b(a1(a1(b(a(b(a1(a1(a1(b(a1(a1(a1(b(a1(a1(x))))))))))))))))) --> b(x)
   derived from: 70154 and 71665
New Rule (72007): a1(b(a(b(a1(a1(a1(b(a1(a1(a1(b(a1(a1(x))))))))))))))) --> a(x)
   derived from: 605 and 71673
New Rule (72125): b(a(b(a1(a1(a1(b(a1(a1(a1(b(a1(a1(x))))))))))))) --> a(a(x))
   derived from: 20 and 72007
New Rule (72274): b(a(b(a1(a1(a1(b(a1(a1(a1(b(a1(x))))))))))))) --> a(a(a(x)))
   derived from: 19 and 72125
New Rule (72425): b(a(b(a1(a1(a1(b(a1(a1(a1(b(x))))))))))) --> a(a(a(a(x))))
   derived from: 19 and 72274
New Rule (72609): a(a(b(a1(a1(b(a(b(b(x))))))))) --> b(a(b(a1(a1(a1(b(a1(a1(x))))))))))
   derived from: 605 and 72425
New Rule (72794): a(b(a1(a1(b(a(b(b(x))))))))) --> a1(b(a(b(a1(a1(a1(b(a1(a1(x))))))))))
   derived from: 19 and 72609
New Rule (72965): b(a1(a1(b(a(b(b(x))))))) --> a1(a1(b(a(b(a1(a1(a1(b(a1(a1(x))))))))))
   derived from: 19 and 72794
New Rule (605): b(a1(a1(a1(a1(b(a(b(a1(a1(a1(b(a1(a1(x))))))))))))))) --> a(x)
```

```
New Rule (73297): b(a1(a1(a1(a1(b(a(b(a1(a1(a1(b(a1(x)))))))))))) --> a(a(x))
   derived from: 19 and 605
New Rule (73434): b(a1(a1(a1(a1(b(a(b(a1(a1(a1(b(x)))))))))))) --> a(a(a(x)))
   derived from: 19 and 73297
New Rule (73436): a1(b(a(b(a1(a1(a1(b(x))))))))) --> b(a1(a1(a1(a1(b(a(b(x))))))))
   derived from: 73434 and 73434
New Rule (73620): b(a1(a1(a1(b(a1(a1(a1(a1(b(a(b(x)))))))))))) --> a(a(a(x)))
   derived from: 72425 and 73436
New Rule (73780): b(a1(a1(a1(a1(b(a(a(a(a(x)))))))))) --> a1(b(a(b(x))))
   derived from: 73436 and 73620
New Rule (74058): a1(b(a(b(a1(x))))) --> b(a1(a1(a1(a1(b(a(a(a(x))))))))
   derived from: 20 and 73780
New Rule (74157): b(a(b(a1(x)))) --> a(b(a1(a1(a1(a1(b(a(a(a(x))))))))))
   derived from: 20 and 74058
New Rule (74171): a(a(b(b(b(a1(x)))))) --> a1(a1(a1(a1(b(a1(b(a(x))))))))
   derived from: 66166 and 74157
New Rule (74368): a(b(b(b(a1(x))))) --> a1(a1(a1(a1(a1(b(a1(b(a(x))))))))
   derived from: 19 and 74171
New Rule (74471): b(b(b(a1(x)))) --> a1(a1(a1(a1(a1(a1(b(a1(b(a(x))))))))))
   derived from: 19 and 74368
New Rule (74583): b(b(b(x))) --> a1(a1(a1(a1(a1(a1(b(a1(b(a(a(x))))))))))
   derived from: 19 and 74471
New Rule (73961): b(a1(a1(a1(b(a1(b(b(a1(b(x))))))))) --> a(a(a(b(a(x)))))
   derived from: 488 and 73620
New Rule (74049): a(a(b(a1(a1(a1(b(a(a(a(a(x))))))))))) --> a(a(a(b(b(x)))))
   derived from: 1219 and 73780
New Rule (74882): a(b(a1(a1(a1(b(a(a(a(a(x))))))))) --> a(a(b(b(x))))
   derived from: 19 and 74049
New Rule (74925): b(a1(a1(a1(b(a(a(a(a(x))))))))) --> a(b(b(x)))
   derived from: 19 and 74882
New Rule (74984): a(b(b(a1(x)))) --> b(a1(a1(a1(b(a(a(a(x))))))))
   derived from: 20 and 74925
New Rule (75072): b(b(a1(x))) --> a1(b(a1(a1(a1(b(a(a(a(x))))))))
   derived from: 19 and 74984
New Rule (75077): a1(b(a(a(b(a1(b(x)))))))) --> b(a1(b(a(b(x)))))
   derived from: 73780 and 75072
New Rule (75229): b(a(a(b(a1(b(x))))))) --> a(b(a1(b(a(b(x))))))
   derived from: 20 and 75077
New Rule (75074): a1(b(a1(a1(a1(b(a(b(a(a(a(a(x))))))))))) --> b(a(b(b(x))))
   derived from: 74925 and 75072
New Rule (75364): b(a1(a1(a1(b(a(b(a(a(a(a(x))))))))))) --> a(b(a(b(b(x)))))
   derived from: 20 and 75074
New Rule (74934): a(a(b(a1(a1(a1(a1(a1(b(a(a(a(a(x))))))))))))) --> b(b(a(x)))
   derived from: 71175 and 74925
New Rule (75458): a(a(b(a1(a1(a1(a1(a1(b(a(a(a(x))))))))))))) --> b(b(x))
   derived from: 20 and 74934
New Rule (75505): a(b(a1(a1(a1(a1(a1(b(a(a(a(x))))))))))) --> a1(b(b(x)))
   derived from: 19 and 75458
New Rule (75547): b(a1(a1(a1(a1(a1(b(a(a(a(x))))))))) --> a1(a1(b(b(x))))
   derived from: 19 and 75505
New Rule (75553): a(b(a1(a1(a1(a1(b(a1(b(a(a(a(x))))))))))) --> b(a1(b(b(x))))
   derived from: 74157 and 75547
New Rule (75636): b(a1(a1(a1(a1(b(a1(b(a(a(a(x))))))))))) --> a1(b(a1(b(b(x)))))
   derived from: 19 and 75553
New Rule (75551): a1(b(a1(a1(a1(b(a1(b(a(a(a(x))))))))))) --> b(a1(a1(b(b(x)))))
   derived from: 75072 and 75547
New Rule (75711): b(a1(a1(a1(b(a1(b(a(a(a(x))))))))))) --> a(b(a1(a1(b(b(x))))))
   derived from: 20 and 75551
New Rule (75369): b(a(a(b(a(b(b(x)))))))) --> a(b(a1(a1(a1(b(a1(a1(b(a(a(x))))))))))
   derived from: 74157 and 75364
New Rule (75367): a1(b(a1(a1(b(a1(a1(b(a(a(x))))))))))) --> a(b(a1(a1(b(a1(a1(b(x))))))))
   derived from: 75072 and 75364
New Rule (75902): b(a1(a1(b(a1(a1(b(a(a(x))))))))) --> a(a(b(a1(a1(b(a1(a1(b(x))))))))
   derived from: 20 and 75367
New Rule (75953): a(a(b(a1(a1(b(a1(a1(b(a1(x))))))))))) --> b(a1(a1(b(a1(a1(b(a(x))))))))
```

```
   derived from: 20 and 75902
New Rule (76032): a(a(b(a1(a1(a1(b(a(b(a(x)))))))))) --> b(a1(a1(b(a1(a1(x))))))
   derived from: 5094 and 75953
New Rule (76067): a(a(a(b(a(b(b(x)))))))) --> b(a1(a1(b(a(x)))))
   derived from: 75364 and 76032
New Rule (76229): a(a(b(a(b(b(x)))))) --> a1(b(a1(a1(b(a(x))))))
   derived from: 19 and 76067
New Rule (76329): a(b(a(b(b(x))))) --> a1(a1(b(a1(a1(b(a(x)))))))
   derived from: 19 and 76229
New Rule (76425): b(a(b(b(x)))) --> a1(a1(a1(b(a1(a1(b(a(x))))))))
   derived from: 19 and 76329
New Rule (76511): b(a1(a1(b(a1(a1(b(a(x))))))))) --> a(b(a1(a1(b(a1(a1(b(x))))))))
   derived from: 336 and 76425
New Rule (76567): a(b(a1(a1(b(a1(a1(b(a1(x))))))))) --> b(a1(a1(b(a1(a1(b(x)))))))
   derived from: 20 and 76511
New Rule (76647): b(a1(a1(b(a1(a1(b(a1(x))))))))) --> a1(b(a1(a1(b(a1(a1(b(x))))))))
   derived from: 19 and 76567
New Rule (76122): a(a(b(a1(a1(a1(b(a(b(x))))))))) --> b(a1(a1(b(a1(a1(a1(x)))))))
   derived from: 20 and 76032
New Rule (76805): a(b(a1(a1(a1(b(a(b(x))))))))) --> a1(b(a1(a1(b(a1(a1(a1(x))))))))
   derived from: 19 and 76122
New Rule (76888): b(a1(a1(a1(b(a(b(x)))))))) --> a1(a1(b(a1(a1(b(a1(a1(a1(x))))))))
   derived from: 19 and 76805
New Rule (75369): a(b(a1(a1(a1(b(a1(a1(b(a(a(x))))))))))) --> a1(a1(a1(b(a1(a1(a1(b(a1(x))))))))
New Rule (75233): a(b(a1(b(a(a(b(b(x))))))))) --> b(a1(a1(a1(a1(b(a1(b(a(a(x))))))))))
   derived from: 74925 and 75229
New Rule (77088): b(a1(b(a(a(b(b(x)))))))) --> a1(b(a1(a1(a1(a1(b(a1(b(a(a(x))))))))))
   derived from: 19 and 75233
New Rule (75088): a1(b(a1(a1(a1(b(a(a(a(b(a1(x))))))))))) --> a1(a1(a1(a1(a1(a1(b(a1(a1(b(a(x))))))))))
   derived from: 68499 and 75072
New Rule (74930): a(b(a1(a1(a1(a1(b(a(b(a(a(a(a(x)))))))))))))) --> b(a(a(b(b(x)))))
   derived from: 74157 and 74925
New Rule (77259): b(a1(a1(a1(a1(b(a(b(a(a(a(a(x))))))))))))) --> a1(b(a(a(b(b(x))))))
   derived from: 19 and 74930
New Rule (77272): b(a1(a1(a1(a1(b(a(a(b(b(x)))))))))) --> a(a(a(a(a(a(a(x)))))))
   derived from: 73620 and 77259
New Rule (14189): b(a1(a1(a1(a1(b(a(a(b(a(b(a(x))))))))))))
                  --> a1(a1(a1(a1(a1(a1(b(a(a(b(a1(a1(b(a(x))))))))))))))
New Rule (77316): b(a1(a1(a1(a1(b(a(a(b(a1(a1(b(b(x))))))))))))) --> a(a(b(a(a(a(x))))))
   derived from: 75547 and 77272
New Rule (77467): a(a(b(a(a(a(a(a(b(x))))))))) --> a(a(a(a(b(a1(a1(x)))))))
   derived from: 71665 and 77316
New Rule (77608): a(b(a(a(a(a(a(b(x))))))))) --> a(a(a(b(a1(a1(x))))))
   derived from: 19 and 77467
New Rule (77694): b(a(a(a(a(a(b(x)))))))) --> a(a(b(a1(a1(x)))))
   derived from: 19 and 77608
New Rule (77725): b(a(a(a(a(a(a(b(b(x))))))))) --> b(b(a(x)))
   derived from: 74925 and 77694
New Rule (77789): b(a1(b(b(a(a(x)))))) --> a(a(b(a1(a1(b(a(b(x))))))))
   derived from: 109 and 77694
New Rule (77714): b(a(a(b(a1(a1(b(a(x))))))))) --> a(a(b(a1(b(b(x))))))
   derived from: 76425 and 77694
New Rule (77830): a1(b(a(b(a(a(b(b(x))))))))) --> b(a1(a1(a1(a1(b(b(a(x))))))))
   derived from: 73780 and 77725
New Rule (78066): b(a(b(a(a(b(b(x)))))))) --> a(b(a1(a1(a1(a1(b(b(a(x))))))))
   derived from: 20 and 77830
New Rule (77794): a(a(b(a(a(a(a(b(b(x))))))))) --> a(a(b(a1(a1(b(a(x)))))))
   derived from: 77694 and 77725
New Rule (78261): a(b(a(a(a(a(b(b(x))))))))) --> a(b(a1(a1(b(a(x))))))
   derived from: 19 and 77794
New Rule (78354): b(a(a(a(a(b(b(x)))))))) --> b(a1(a1(b(a(x)))))
   derived from: 19 and 78261
New Rule (78452): b(a1(a1(b(b(a(a(x)))))))) --> b(a1(a1(b(a(a(b(x)))))))
   derived from: 109 and 78354
New Rule (78507): b(a1(a1(b(a(a(b(a1(x))))))))) --> b(a1(a1(b(b(a(x))))))
```

```
   derived from: 20 and 78452
New Rule (78548): a(b(a(a(b(a1(x))))))  --> a(b(b(a(x))))
   derived from: 73620 and 78507
New Rule (78664): b(a(a(b(a1(x)))))  --> b(b(a(x)))
   derived from: 19 and 78548
New Rule (78737): b(b(a(a(x))))  --> b(a(a(b(x))))
   derived from: 19 and 78664
New Rule (78748): b(a(a(b(a(a(a(b(x))))))))  --> b(b(x))
   derived from: 77694 and 78737
New Rule (75229): a(b(a1(b(a(b(x))))))  --> a1(a1(a1(a1(a1(a1(b(a(a(b(x))))))))))
New Rule (78792): b(a(a(a(a(b(a1(a1(a1(a1(x))))))))))  --> b(b(x))
   derived from: 77694 and 78748
New Rule (78997): b(a(a(a(a(b(a1(a1(a1(x))))))))  --> b(b(a(x)))
   derived from: 19 and 78792
New Rule (79057): b(a(a(a(a(b(a1(a1(x))))))))  --> b(a(a(b(x))))
   derived from: 19 and 78997
New Rule (79126): b(a(a(a(a(b(a1(x)))))))  --> b(a(a(b(a(x)))))
   derived from: 19 and 79057
New Rule (79199): b(a(a(b(a(a(x))))))  --> b(a(a(a(a(b(x))))))
   derived from: 19 and 79126
New Rule (79206): b(a(a(b(b(a(x))))))  --> b(a1(a1(b(x))))
   derived from: 78664 and 79199
New Rule (79306): b(a(a(b(b(x)))))  --> b(a1(a1(b(a1(x)))))
   derived from: 20 and 79206
New Rule (78748): b(a(a(a(a(b(a(b(x))))))))  --> b(b(x))
New Rule (78760): a(a(a(b(a(a(x))))))  --> a(a(a(a(a(b(x))))))
   derived from: 73620 and 78737
New Rule (79524): a(a(b(a(a(x)))))  --> a(a(a(a(b(x)))))
   derived from: 19 and 78760
New Rule (79573): a(b(a(a(x))))  --> a(a(a(b(x))))
   derived from: 19 and 79524
New Rule (79613): b(a(a(x)))  --> a(a(b(x)))
   derived from: 19 and 79573
New Rule (79631): a(a(b(a1(x))))  --> b(a(x))
   derived from: 20 and 79613
New Rule (79651): a(b(a1(x)))  --> a1(b(a(x)))
   derived from: 19 and 79631
New Rule (79675): b(a1(x))  --> a1(a1(b(a(x))))
   derived from: 19 and 79651
New Rule (74157): a1(a1(b(a(b(a(x))))))  --> a1(b(b(a(x))))
New Rule (79684): b(a(b(x)))  --> a(b(b(x)))
   derived from: 79613 and 74157
New Rule (79306): a1(a1(a1(a1(b(b(a(x)))))))  --> a1(a1(a1(b(b(x)))))
New Rule (79701): a1(a1(a1(b(b(a(x))))))  --> a1(a1(b(b(x))))
   derived from: 20 and 79306
New Rule (79707): a1(a1(b(b(a(x)))))  --> a1(b(b(x)))
   derived from: 20 and 79701
New Rule (79713): a1(b(b(a(x))))  --> b(b(x))
   derived from: 20 and 79707
New Rule (79719): b(b(a(x)))  --> a(b(b(x)))
   derived from: 20 and 79713
New Rule (77694): a(a(a(a(b(b(x)))))))  --> b(x)
New Rule (79728): b(a(x))  --> a(b(x))
   derived from: 79719 and 77694
New Rule (79738): a(a(a(a(b(b(x))))))  --> a1(b(x))
   derived from: 19 and 77694
New Rule (79750): a(a(a(b(b(x)))))  --> a1(a1(b(x)))
   derived from: 19 and 79738
New Rule (79757): a(a(b(b(x))))  --> a1(a1(a1(b(x))))
   derived from: 19 and 79750
New Rule (79764): a(b(b(x)))  --> a1(a1(a1(a1(b(x)))))
   derived from: 19 and 79757
New Rule (79771): b(b(x))  --> a1(a1(a1(a1(a1(b(x))))))
   derived from: 19 and 79764
New Rule (73620): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(
```

```
                    a1(a1(b(x))))))))))))))))))))) --> a(a(a(x)))
New Rule (79777): a(a(a(b(x)))) --> a1(a1(x))
   derived from: 79771 and 73620
New Rule (79787): a(a(b(x))) --> a1(a1(a1(x)))
   derived from: 19 and 79777
New Rule (79793): a(b(x)) --> a1(a1(a1(a1(x))))
   derived from: 19 and 79787
New Rule (79799): b(x) --> a1(a1(a1(a1(a1(x)))))
   derived from: 19 and 79793
New Rule (77272): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(
                     a1(a1(x))))))))))))))))))) --> a(a(a(a(a(a(a(x)))))))
New Rule (79308): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x))))))))))))))))))))
                  --> a(a(a(a(a(a(a(a(x)))))))
   derived from: 0 and 0
New Rule (69317): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x))))))))))))))))))
                  --> a(a(a(a(a(a(a(a(a(a(x)))))))))
   derived from: 0 and 0
New Rule (78689): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x))))))))))))))))
                  --> a(a(a(a(a(a(a(a(a(a(x)))))))))
   derived from: 0 and 78664
New Rule (78670): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x))))))))))))))))
                  --> a(a(a(a(a(a(a(a(a(a(x)))))))))
   derived from: 77789 and 78664
New Rule (77932): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x))))))))))))))
                  --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))
   derived from: 20 and 77789
New Rule (77833): a1(a1(a1(a1(a1(a1(a1(a1(a1(a1(x)))))))))
                  --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))
   derived from: 0 and 0
New Rule (76453): a1(a1(a1(a1(a1(a1(a1(a1(a1(x)))))))))
                  --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))
   derived from: 0 and 0
New Rule (75949): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                    a(a(a(a(a(a(a(x)))))))))))))))))))))))))))))))))))
                  --> a(a(a(a(a(a(a(a(x)))))))))
   derived from: 0 and 0
New Rule (75908): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                    a(a(a(a(a(a(x)))))))))))))))))))))))))))))))))))) --> a(a(a(a(a(a(x)))))))
   derived from: 0 and 0
New Rule (1938): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                   a(a(a(x)))))))))))))))))))))))))))))))))) --> a(a(a(a(x))))
New Rule (3659): a1(a1(a1(x)))
                 --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))))
New Rule (1205): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                   a(a(x)))))))))))))))))))))))))))))))) --> a(a(a(x)))
   derived from: 0 and 0
New Rule (19839): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                    a(x)))))))))))))))))))))))))))))))) --> a(a(x))
   derived from: 3659 and 0
New Rule (19649): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                    a(x)))))))))))))))))))))))))))))))) --> a(x)
   derived from: 3659 and 0
New Rule (27209): a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                    a(x)))))))))))))))))))))))))))))))) --> x
   derived from: 0 and 0
New Rule (26891): a1(a1(x))
                  --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))))
   derived from: 0 and 0
New Rule (4130): a1(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(
                             a(a(a(a(a(a(a(a(x)))))))))))))))))))))))))))))))
   derived from: 1134 and 0

   Result:
a1(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))))
a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))) --> x
```

```
b(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))
b1(x) --> a(a(a(a(x))))
9(x) --> a(a(a(x)))
c(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))))))
d1(x) --> a(a(a(a(a(a(a(a(x)))))))))
d(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))
e1(x) --> a(a(a(a(a(a(a(a(a(a(x)))))))))
e(x) --> a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))
f1(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))
f(x) --> a(a(a(a(a(a(a(x)))))))
1(x) --> a(a(a(a(a(x)))))
g(x) --> a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(a(x)))))))))))))))))
```

Danksagung

Zu Märdistan, im Walde von Kulub,
Liegt einsam, tief versteckt die Geisterschmiede.
Nicht schmieden Geister; nein, man schmiedet sie!

Karl May

Wegen des persönlichen und weniger fachlichen Charakters einer Danksagung verfasse ich diese in meiner deutschen Muttersprache, für die ich weitaus besser beurteilen kann, was ich überhaupt ausdrücke und ob es das trifft, was ich sagen will.

Das Sprechen einer gemeinsamen Sprache, ein gemeinsamer Hintergrund und ein gemeinsames Arbeitsziel müssen allerdings noch lange nicht zum einander Verstehen führen, wie ich in verschiedenen Gesprächen und Auseinandersetzungen mit meinem Doktorvater Jörg Siekmann erfahren mußte. Ich muß ihm dafür danken, daß er trotz dieses gegenseitigen Unverständnisses und diverser Streits diese Arbeit bis zu ihrem Abschluß mit allen ihm zur Verfügung stehenden Kräften unterstützt hat. Seine Ratschläge beschränkten sich leider auf die Darstellung und Präsentation der Arbeit und zahlreiche Nachfragen nach ihrem Fortgang.

Durch meine Einstellung im SFB 314 in Kaiserslautern und später als freier Mitarbeiter am DFKI in Saarbrücken wurde diese Arbeit überhaupt erst ermöglicht.

Zu danken habe ich ganz besonders Gebhard Przyrembel, ohne dessen unermüdliche Versuche, unsere Rechnermenagerie in Zaum zu halten, kein einziges Programm unserer Arbeitsgruppe laufen würde. Dies war für meine Arbeit um so wichtiger, da sie sehr praktisch orientiert ist und bis in die Untiefen des Sparc-Assemblers führte. Gebhard stand mir zu jeder Zeit von morgens bis nachts mit Rat und Tat, manchmal auch mit dem Tischtennisschläger, zur Verfügung. In Saarbrücken übernahm Dan Nesmith die undankbare Aufgabe der Rechnerbetreuung mit gleichem Engagement.

Manfred Kerber habe ich nicht nur für das Korrekturlesen der Arbeit zu danken, sondern auch für die jahrelange fachkundige Benutzung des MKRP-Systems. Ohne ihn wären viele Fehler nicht gefunden und damit auch nicht beseitigt worden. Seine Beispiele brachten mir sehr viel Verständnis für die Einsetzbarkeit von automatischen Beweisern, und sein politisches Engagement erweckte bei mir die Erkenntnis, solche Computerprogramme eben nicht immer einsetzen zu dürfen. Dieser durchaus rationale Blick über die engen Grenzen unseres Fachgebietes hinaus hielt meine Motivation aufrecht, die Arbeit fortzusetzen und zu beenden. Die Zusammenarbeit mit Manfred im FiFF und bei der Veranstaltung mehrerer Seminare schärfte vielleicht auch bei anderen den Blick für die gesellschaftlichen Auswirkungen der Informatik und der KI im Besonderen.

Für zahlreiche Anregungen und konstruktive Verbesserungsvorschläge habe ich meinen Kollegen Jürgen Cleve, Jörg Denzinger, Manfred Kerber und Christoph Lingenfelder, sowie Klaus Madlener als zweitem Gutachter zu danken, Christoph danke ich auch für die Zusammenarbeit bezüglich des Abschnitts 6.6.

Jörn Richts, „meinem“ Diplomanden und HiWi habe ich für die Implementierung des Narrowingbeweisers und die Wartung der Theorieunifikationsalgorithmen in HADES zu danken, sowie für die Arbeit

beim Entwurf von Ω-MKRP. Ohne sein Engagement hätte sich der Abschluß dieser Arbeit noch weiter verzögert.

Norbert Eisinger danke ich für die intensive Betreuung beim Einstieg in die Arbeitsgruppe.

Auf dem langen und beschwerlichen Weg von meiner Aufnahme in die AG Siekmann bis zu diesem krönenden Abschluß lernte ich viele Menschen kennen, die mich in meiner Sicht der Welt bestärkten, selbst wenn sie sich mit missionarischem Eifer bemühten, mein Weltbild zum Einsturz zu bringen. Sie haben letztendlich dazu beigetragen, dieses zu stabilisieren. Bedauerlicherweise ist man nicht hinreichend immun gegen diverse Paradoxien des Wissenschaftsbetriebs, der nun einmal die polarisierende einer differenzierenden Betrachtungsweise und Darstellung vorzieht. Einige solcher von mir gar nicht erwünschten Polarisierungen finden sich sicher auch in dieser Arbeit.

Ich möchte mich bei Jörg Denzinger, Detlef Fehrer, Horst Gerlach, Manfred Kerber, Michael Kohlhase, Andreas Nonnengart, Gebhard Przyrembel, Christoph Weidenbach und allen nicht so oft Belästigten auch für ihr nachgeradezu unermüdliches Zuhören bedanken, wenn ich von meinen Frustrationen an der Wissenschaft berichtete.

Weiteren Dank schulde ich unseren Sekretärinnen Dorothea Kilgore und Maria Pfeiffer in Kaiserslautern sowie Agnes Back in Saarbrücken für die Verwaltungarbeit im Hintergrund, ohne die man sich leicht in den Wirren der Bürokratie verirrt.

Horst Gerlach und John Kalman danke ich für Ratschläge und neue Testbeispiele.

Ich möchte auch denen danken, die in vielen Tee-, Kaffee-, Keks- und Kuchenpausen eine menschlich warme Atmosphäre schufen, die wohltuend von derjenigen in Arbeitsgruppenbesprechungen abwich. Ich möchte hier ohne vollständig sein zu wollen Bettina und Sonja Anslinger, Kerstin und Klaus Becker, Detlef und Judith Fehrer, Claudia Graf, Monika Keil, Andreas Nonnengart, Gebhard Przyrembel, Rosa Ruggeri, Petra Stoll, Martin Strobl und Christoph Weidenbach nennen. Ohne die Teepausen als gesellschaftliches Ereignis wäre wissenschaftliches Arbeiten für mich nicht möglich gewesen.

Bei all denjenigen, die ich in dieser Danksagung vergessen habe und bei allen, denen ich unabsichtlich auf die Füße getreten bin, bitte ich hiermit aufrichtig um Vergebung.

Bibliography

[AGM90] Hitoshi Aida, Joseph Goguen, and José Meseguer. Compiling concurrent rewriting onto the rewrite rule machine. Report SRI-CSL-90-03R, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, February, Rev. December 1990.

[AHU83] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Computer Science and Information Processing. Addison Wesley Publishing Company, Reading, Massachusetts, USA, 1983.

[ALM90] Hitoshi Aida, Sani Leinwand, and José Meseguer. Architectural design of the rewrite rule machine ensemble. Report SRI-CSL-90-17, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, December 1990.

[And70] Robert Anderson. Completeness results for E-resolution. In *Spring Joint Computer Conference, American Federation of Information Processing Societies (AFIPS), Conference Proceedings, Volume 36*, pages 653–656, Atlanta City, New Jersey, May 1970. AFIPS Press, 210 Summit Avenue, Montvale, New Jersey 07645.

[AO83] Grigorios Antoniou and Hans Jürgen Ohlbach. Terminator. In Alan Bundy, editor, *Proceedings 8^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 916–919, Karlsruhe, Germany, August 1983. William Kaufmann, Inc., 95 First Street, Los Altos, California 94022, USA.

[AS84] Harold Abelson and Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, Massachusetts, USA, 1984.

[Bau90] Peter Baumgartner. Combinig Horn clause logic with rewrite rules. Report FKI-131-90, Institut für Informatik, Technische Universität München, Arcisstr. 21, 8000 München 2, Germany, May 1990.

[BB89] Wray L. Buntine and Hans-Jürgen Bürckert. On solving equations and disequations. SEKI-Report SR-89-03, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1989.

[BBB+84] Susanne Biundo, Karl Hans Bläsius, Hans-Jürgen Bürckert, Norbert Eisinger, Alexander Herold, Thomas Käufl, Christoph Lingenfelder, Hans Jürgen Ohlbach, Manfred Schmidt-Schauß, Jörg H. Siekmann, and Christoph Walther. The Markgraf Karl Refutation Procedure. SEKI-MEMO MK-84-01, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, Institut für Informatik I, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe 1, Germany, 1984.

[BBH91] Richard Barnett, David Basin, and Jane Hesketh. A recursion planning analysis of inductive completion. D.A.I. Research Report 518, Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland, 1991.

[BCM89] Pier Giorgio Bosco, C. Cecchi, and Corrado Moiso. An extension of WAM for K-LEAF: A WAM-based compilation of conditional narrowing. In *Proceedings 6^{th} International Conference on Logic Programming*, 1989.

[BDJ79] Daniel Brand, John A. Darringer, and William H. Joyner. Completeness of conditional reductions. In William H. Joyner, editor, *Proceedings 4^{th} Workshop on Automated Deduction*, pages 36–42, Austin, USA, February 1979. Aso published as Report, IBM Thomas J. Watson Research Center, Yorktown Heights, 1978.

[BDP87] Leo Bachmair, Nachum Dershowitz, and David Plaisted. Completion without failure. In H. Ait-Kaci and M. Nivat, editors, *Conference on Resolution of Equations in Algebraic Structures (CREAS)*, Lakaway, Texas, USA, 1987.

[BDP89] Leo Bachmair, Nachum Dershowitz, and David Plaisted. *Completion without Failure*, volume 2 of *Resolution of Equations in Algebraic Structures*. Academic Press, New York, USA, 1989.

[BG85] M. Bidoit and M. C. Glaudel. PLUSS: Proposition pour une langage de spécification structurée. *Bicre + Globèle 45, France*, 1985.

[BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In Mark E. Stickel, editor, *Proceedings 10^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 427–441, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[BGLS92] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Deepak Kapur, editor, *Proceedings 11^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 607*, pages 462–476, Saratoga Springs, New York, USA, June 1992. Springer-Verlag, Berlin, Germany.

[BGM88] Pier Giorgio Bosco, Elio Giovannetti, and Corrado Moiso. Narrowing vs. SLD-resolution. *Theoretical Computer Science, North Holland, Elsevier Science Publishers B.V.*, 59:3–23, 1988.

[BH91] Maria Paola Bonacina and Jieh Hsiang. Towards a foundation of completion procedures as semidecision procedures. FTP-Report, Department of Computer Science, SUNY at Stony Brook, Stony Brook, New York 11794-4400, USA, 1991.

[Bib82] Wolfgang Bibel. *Automated Theorem Proving*. Vieweg Verlag, Wiesbaden, Germany, 1982.

[Blä86] Karl Hans Bläsius. *Equality Reasoning Based on Graphs*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1986. Also published as SEKI-Report SR-87-01, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany.

[BM79] R. S. Boyer and J Strother Moore. *A Computational Logic*. Academic Press, London, England, 1979.

[Bra75] Daniel Brand. Proving theorems with the modification method. *SIAM (Society for Industrial and Applied Mathematics) Journal of Computing*, 4(4):412–430, December 1975.

[Bru75] M. Bruynooghe. The inheritance of links in a connection graph. Report CW2, Applied Mathematics and Programming Division, Katholieke Universiteit Leuven, 1975.

[BS87] Wolfgang Büttner and Helmut Simonis. Embedding Boolean expressions into logic programming. *Journal of Symbolic Computation, Academic Press, Inc., London, England*, 4:191–205, 1987.

[BS92] Franz Baader and Klaus U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In Deepak Kapur, editor, *Proceedings 11th International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 607*, pages 50–65, Saratoga Springs, New York, USA, June 1992. Springer-Verlag, Berlin, Germany.

[Buc85] Bruno Buchberger. History and basic features of the critical-pair/completion approach. *Dijon, France*, 1985.

[Bun74] Alan Bundy. Analysing mathematical proofs (or reading between the lines). D.A.I. Research Report 2, Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland, 1974.

[Bun83] Alan Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, London, England, 1983.

[Bun89] Alan Bundy. A science of reasoning. D.A.I. Research Paper 445, Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland, October 1989.

[BvHSI90] Alan Bundy, Frank van Harmelen, Alan Smaill, and Andrew Ireland. Extension to the ripling-out tactic for guiding inductive proofs. In Mark E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 132–146, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[CD85] R. J. Cunningham and A. J. J. Dick. Rewrite systems on a lattice of types. *Acta Informatica, Springer-Verlag, Berlin, Germany*, 22:149–169, 1985.

[CL91] Manuel M. T. Chakravarty and Hendrik C. R. Lock. The implementation of lazy narrowing: The jump machine. Arbeitspapier der GMD 530, Gesellschaft für Mathematik und Datenverarbeitung mbH, Schloß Birlinghoven, Postfach 1240, 5205 Sankt Augustin 1, Germany, April 1991.

[Coh87] Anthony G. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 3(2):113–200, 1987.

[CR79] S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.

[CR89] Bernhard Crone-Rawe. Unification algorithms for Boolean rings. SEKI-Working Paper SWP-89-01, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1989.

[Dar68] J. L. Darlington. Automatic theorem proving with equality substitution and mathematical induction. *Machine Intelligence, Edinburgh University Press, 22 George Street, Edinburgh, Scotland*, 3:113–127, 1968.

[Den91] Jörg Denzinger. Distributed knowledge-based deduction using the team work method. SEKI-Report SR-91-12, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1991.

[Der87] Nachum Dershowitz. Rewriting systems. Draft, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA, 1987.

[Der91] Nachum Dershowitz. Ordering-based strategies for Horn clauses. In John Mylopoulos and Ray Reiter, editors, *Proceedings 12^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 118–124, Sydney, Australia, 1991. Morgan Kaufmann Publishers, Inc., 2929 Campus Drive San Mateo, California 94403, USA.

[DG89] John Darlington and Yi-Ke Guo. Narrowing and unification in functional programming - an evaluation mechanism for absolute set abstraction. In Nachum Dershowitz, editor, *Proceedings 3^{rd} Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 355*, pages 92–108, Chapel Hill, North Carolina, USA, April 1989. Springer-Verlag, Berlin, Germany.

[Dic85] A. J. J. Dick. Eril – equational reasoning, an interactive laboratory. In Bruno Buchberger, editor, *Proceedings European Computer Algebra Conference, EUROCAL'85, volume 2, Lecture Notes in Computer Science (LNCS) 204*, Linz, Austria, April 1985. Springer-Verlag, Berlin, Germany.

[Dig79] Vincent J. Digricoli. Resolution by unification and equality. In William H. Joyner, editor, *Proceedings 4^{th} Workshop on Automated Deduction*, pages 43–52, Austin, USA, February 1979.

[Dig81] Vincent J. Digricoli. The efficacy of RUE resolution, experimental results and heuristic theory. In Ann Drinan, editor, *Proceedings 7^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 539–547, Vancouver, Canada, 1981. William Kaufmann, Inc., 95 First Street, Los Altos, California 94022, USA.

[Dig85] Vincent J. Digricoli. The management of heuristic search in Boolean experiments with RUE resolution. In Aravind Joshi, editor, *Proceedings 9^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1154–1161, Los Angeles, USA, 1985.

[DJ90] Daniel J. Dougherty and Patricia Johann. An improved general E-unification method. In Mark E. Stickel, editor, *Proceedings 10^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 261–275, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[DP85] Nachum Dershowitz and David A. Plaisted. Logic programming cum applicative programming. In *1985 Symposium on Logic Programming*, pages 54–66, Boston, Massachusetts, USA, 1985. IEEE Computer Society Press, Washington D.C. 20036-1903, USA.

[EFT86] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik*. Wissenschaftliche Buchgesellschaft, Darmstadt, Germany, 2^{nd} edition, 1986. The first edition was published 1978.

[Eis88] Norbert Eisinger. *Completeness, Confluence, and Related Properties of Clause Graph Resolution*. PhD thesis, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1988. Also published as SEKI-Report SR-88-07, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1988.

[EOP89] Norbert Eisinger, Hans Jürgen Ohlbach, and Axel Präcklein. Elimination of redundancies in clause sets and clause graphs. SEKI-Report SR-89-10, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, October 1989. Also published as: Reduction Rules for Resolution-Based Systems, in Artificial Intelligence 50 (1991), pages 141–181, Elsevier Science Publishers B.V.

[Fag83] F. Fages. Formes canoniques dans les algèbres booléennes et application à la démonstration automatique en logique de premier ordre. *Thèse de 3ème Cycle, Paris*, 1983.

[Fay79] Michael J. Fay. First order unification in an equational theory. In William H. Joyner, editor, *Proceedings 4th Workshop on Automated Deduction*, pages 161–167, Austin, USA, February 1979.

[FGJM85] Kokichi Futatsugi, Joseph A. Goguen, Jean-Pierre Jouannaud, and José Meseguer. Principles of OBJ2. In *Proceedings 12th ACM Symposium on Principles of Programming Languages*, 1985.

[FHS91] Ulrich Furbach, Steffen Hölldobler, and Joachim Schreiber. Linear paramodulation modulo equality. Report FKI-111-89, Institut für Informatik, Technische Universität München, Arcisstr. 21, 8000 München 2, Germany, December 1991.

[Fri84a] Laurent Fribourg. A narrowing procedure for theories with constructors. In Robert E. Shostak, editor, *Proceedings 7th International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 170*, pages 259–301, Napa, California, USA, May 1984. Springer-Verlag, Berlin, Germany.

[Fri84b] Laurent Fribourg. Oriented equational clauses as a programming language. *Journal of Logic Programming, Elsevier Science Publishing Co., Inc., 52 Vanderbilt Avenue, New York 10017, USA*, 2:165–177, 1984.

[Fri85a] Laurent Fribourg. Handling function definitions through innermost superposition and rewriting. In Jean-Pierre Jouannaud, editor, *Proceedings 1st Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 202*, pages 325–344, Dijon, France, May 1985. Centre de Recherche en Informatique de Nancy, Springer-Verlag, Berlin, Germany.

[Fri85b] Laurent Fribourg. SLOG: A logic programming language interpreter based on clausal superposition and rewriting. In *1985 Symposium on Logic Programming*, pages 172–184, Boston, Massachusetts, USA, 1985. IEEE Computation Society Press, 1730 Massachusetts Avenue, Washington, D.C. 20036-1903, USA.

[Fuc90] Matthias Fuchs. Implementation vo Heuristiken zur Behandlung von Gleichheitsinferenzen im Theorembeweiser EQTHEOPOGLES. Projektarbeit, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 6750 Kaiserslautern, Germany, July 1990.

[FW87] Jon Fairbairn and Stuart Wray. TIM – A simple machine to execute supercombinators. In G. Kahn, editor, *Proceedings Conference on Functional Programming Languages and Computer Architecture, Lecture Notes in Computer Science (LNCS) 274*. Springer-Verlag, Berlin, Germany, 1987.

[Gan91] Harald Ganzinger. A completion procedure for conditional equations. *Journal of Symbolic Computation, Academic Press, Inc., London, England*, 11:51–81, 1991.

[Gee92] P. A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings 9th European Conference on Artificial Intelligence (ECAI)*, 1992. Submitted.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift*, 39:572–595, 1935.

[GI88] Jean H. Gallier and T. Isakowitz. Rewriting in order-sorted equational logic. Draft, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia 19104-6389, USA, 1988.

[GJM85] Joseph A. Goguen, Jean-Pierre Jouannaud, and José Meseguer. Operational semantics for order-sorted algebra. In Wilfried Brauer, editor, *Proceedings Automata, Languages and Programming, 12th Colloquium (ICALP), Lecture Notes in Computer Science (LNCS) 194*, pages 221–231, Nafplion, Greece, July 1985. Springer-Verlag, Berlin, Germany.

[GLLO84] John Gabriel, Tim Lindholm, Ewing L. Lusk, and Ross A. Overbeek. A tutorial on the Warren abstract machine. Report ANL-84-84, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, Illinois 60439, USA, 1984.

[GM85] Joseph A. Goguen and José Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In Doug DeGroot and Gary Lindstrom, editors, *Functional and Logic Programming*, pages 295–363. Prentice Hall, Englewood Cliffs, New Jersey 07632, USA, 1985.

[Gog90] Joseph A. Goguen. Semantic specifications for the rewrite rule machine. Technical Report SRI-CSL-90-13, Computer Science Laboratory, SRI International, December 1990.

[GR89] Jean H. Gallier and Stan Raatz. Extending SLD resolution to equational Horn clauses using E-unification. *Journal of Logic Programming, Elsevier Science Publishing Co., Inc., 52 Vanderbilt Avenue, New York 10017, USA*, 6(1-2):3–43, January and March 1989.

[GS86] Jean H. Gallier and Wayne Snyder. A general complete E-unification procedure. Technical report, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia 19104-6389, USA, 1986.

[GS89] Jean H. Gallier and Wayne Snyder. Complete sets of transformations for general E-unification. Report MS-CIS-89-12, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia 19104-6389, USA, December 1989. Also published in Theoretical Computer Science, 1988.

[Her30] Jacques Herbrand. Recherches sur la théorie de la démonstration. *Travaux de la societé des sciences et de lettre de Varsovie, Class III Science mathématique et physique*, 33, 1930.

[HM89] Jochen Hager and Martin Moser. An approach to parallel unification using transputers. In D. Metzing, editor, *Proceedings 13th German Workshop on Artificial Intelligence (GWAI), Informatikfachberichte (IFB) 216*, pages 83–91, Eringerfeld, Germany, September 1989. Springer-Verlag, Berlin, Germany.

[HO80] Gérard Huet and Derek C. Oppen. Equations and rewrite rules: a survey. Technical Report CSL-III, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, 1980.

[Hof88] Thomas Hoffmann. Effizientes AC1-Matching durch Constraint-Propagation. Projektarbeit, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 6750 Kaiserslautern, Germany, August 1988.

[HR86] Jieh Hsiang and Michaël Rusinowitch. A new method for establishing refutational completeness in theorem proving. In Jörg H. Siekmann, editor, *Proceedings 8th International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 230*, pages 141–152, Oxford, England, 1986. Springer-Verlag, Berlin, Germany.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 1133 Avenue of the Americas, New York 10036, USA*, 27(4):798–821, 1980.

[Hul80] Jean-Marie Hullot. Canonical forms and unification. In Wolfgang Bibel and Robert Kowalski, editors, *Proceedings 5th International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 87*, pages 318–334, Les Arcs, France, July 1980. Springer-Verlag, Berlin, Germany.

[Hus85] Heinrich Hussmann. Unification in conditional-equational theories. Report MIP-8502, Fakultät für Mathematik und Informatik, Universität Passau, Postfach 2540, 8390 Passau, January 1985.

[Hut90] Dieter Hutter. Guiding induction proofs. In Mark E. Stickel, editor, *Proceedings 10^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 147–161, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[Jas33] Stanislaw Jaskowski. On the rules of supposition in formal logic. *Studia Logica*, 1, 1933.

[JKK83] Jean-Pierre Jouannaud, Claude Kirchner, and Hélène Kirchner. Incremental construction of unification algorithms in equational theories. In Josep Díaz, editor, *Proceedings Automata, Languages and Programming, 10^{th} Colloquium (ICALP), Lecture Notes in Computer Science (LNCS) 154*, pages 361–373, Barcelona, Spain, July 1983. Springer-Verlag, Berlin, Germany.

[JL87] Jean-Pierre Jouannaud and Pierre Lescanne. Rewriting systems. *Technology and Science of Informatics*, 6(3):181–199, 1987.

[JLM84] Joxan Jaffar, Jean-Louis Lassez, and Michael J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming, Elsevier Science Publishing Co., Inc., 52 Vanderbilt Avenue, New York 10017, USA*, 3:211–223, 1984.

[JW86] Jean-Pierre Jouannaud and B. Waldmann. Reductive conditional term rewriting systems. In *Proceedings 3^{rd} IFIP Conference on Formal Description of Programming Concepts*, Lyngby, 1986.

[Kap84a] Stéphane Kaplan. Conditional rewrite rules. *Theoretical Computer Science, North Holland, Elsevier Science Publishers B.V.*, 33:175–193, 1984.

[Kap84b] Stéphane Kaplan. Fair conditional term rewriting systems: Unification, termination and confluence. *Laboratoire de Recherche en Informatique, Université d'Orsay, France*, 1984.

[Kap87] Stéphane Kaplan. A compiler for conditional term rewriting systems. In Pierre Lescanne, editor, *Proceedings 2^{nd} Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 256*, pages 25–41, Bordeaux, France, May 1987. Springer-Verlag, Berlin, Germany.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[Ker89] Manfred Kerber. Some aspects of analogy in mathematical reasoning. In Klaus P. Jantke, editor, *Analogical and Inductive Inference; International Workshop AII '89, Lecture Notes in Artificial Intelligence (LNAI) 397*, pages 231–242, Reinhardsbrunn Castle, GDR, October 1989. Springer-Verlag, Berlin, Germany.

[Kir85] Claude Kirchner. Méthodes et outils de conception systematique d'algorithmes d'unification dans les théories equationelles. Thèse de Doctorat d'État en Mathématique, Nancy, France, 1985.

[Kir87] Claude Kirchner. Methods and tools for equational unification. In H. Ait-Kaci and M. Nivat, editors, *Conference on Resolution of Equations in Algebraic Structures (CREAS)*, Lakaway, Texas, USA, 1987.

[KK89] Claude Kirchner and Hélène Kirchner. Constrained equational rewriting. In Hans-Jürgen Bürckert and Werner Nutt, editors, *UNIF'89 Extended Abstracts of the 3rd International Workshop on Unification, SEKI-Report SR-89-17*, pages 160–171, Postfach 3049, 6750 Kaiserslautern, Germany, 1989. Fachbereich Informatik, Universität Kaiserslautern.

[Kow75] Robert Kowalski. A proof procedure using connection graphs. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 1133 Avenue of the Americas, New York 10036, USA*, 22(4):572–595, 1975.

[KP92] Manfred Kerber and Axel Präcklein. Tactics for the improvement of problem formulation in resolution-based theorem proving. SEKI-Report SR-92-09 (SFB), Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald 15, 6600 Saarbrücken 11, Germany, 1992. Talk held at the Second International Symposium on Artificial Intelligence and Mathematics in Fort Lauderdale, Florida, USA, 1992.

[Kre89] Per Kreuger. EWAM An extension of WAM to execute functional programs. Technical Report SICS/T89/89016, Swedish Institute of Computer Science, Box 1263, 16428 Kista, Sweden, November 1989.

[Kri90] Stefan Krischer. Vergleich und Optimierung von Narrowing-Strategien. Diplomarbeit, Universität Karlsruhe, 7500 Karlsruhe, Germany, März 1990.

[KZ89] Deepak Kapur and Hantao Zhang. A case study of the completion procedure: Proving ring commutativity problems. *State University of New York at Albany*, 1989.

[Lan75] D. S. Lankford. Canonical inference. Report ATP-32, Department of Computer Science, University of Texas, Austin, Texas 78712, USA, 1975.

[Lan79] D. S. Lankford. Some new approaches to the theory and applications of conditional term rewriting systems. Report, Department of Computer Science, University of Texas, Austin, Texas 78712, USA, 1979.

[Lin90] Christoph Lingenfelder. Transformation and structuring of computer generated proofs. SEKI-Report SR-90-26 (SFB), Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1990.

[LO84] Ewing L. Lusk and Ross A. Overbeek. A short problem set for testing systems that include equality reasoning. Report, Argonne National Laboratory, 1984.

[Lot88] Volkmar Lotz. Heuristische Kontrolle des Aufbaus von Gleichheitsgraphen. Diplomarbeit, Postfach 3049, 6750 Kaiserslautern, Germany, 1988.

[LP91a] Christoph Lingenfelder and Axel Präcklein. Presentation of proofs in an equational calculus. In John Mylopoulos and Ray Reiter, editors, *Proceedings 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 165–170, Sydney, Australia, 1991. Morgan Kaufmann Publishers, Inc., 2929 Campus Drive San Mateo, California 94403, USA. Also published as SEKI-Report SR-90-15 (SFB), Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany.

[LP91b] Christoph Lingenfelder and Axel Präcklein. Proof transformation with built-in equality predicate. In *Proceedings 1st World Conference on the Fundamentals of Artificial Intelligence*, pages 313–321, Paris, France, 1991. Also published as SEKI-Report SR-90-13 (SFB), Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany.

[Mah91] Anne Mahn. Theorievervollständigung mit Constraints. Diplomarbeit, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 6750 Kaiserslautern, Germany, September 1991.

[McC88] William McCune. Challenge equality problems in lattice theory. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 704–709, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[Men87] Elliott Mendelson. *Introduction to Mathematical Logic.* The Wadsworth & Brooks/Cole, Advanced Books and Software, Mathematics Series. Wadsworth, Monterey, California 93940, USA, 3^{rd} edition, 1987. The first edition was published 1964.

[Mes90] José Meseguer. Conditional rewriting logic: Deduction, models and concurrency. CSL Technical Report SRI-CSL-90-14, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, November 1990.

[Mes91] José Meseguer. Conditional rewriting logic as a unified model of concurrency. Technical Report SRI-CSL-91-05, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, February 1991. To appear in Theoretical Computer Science, North Holland, Elsevier Science Publishers B.V., 1992.

[MM82] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

[MMR86] Alberto Martelli, Corrado Moiso, and G. F. Rossi. Lazy unification algorithms for canonical rewrite systems. Report, Dipartimento di Informatica – CSELT, C. so Svizera 185, 10149 Torino, Italia – Via Reiss Romoli 274, 10148 Torino, Italia, 1986.

[Mon55] Richard Montague. On the paradox of grounded classes. *Journal of Symbolic Logic*, 20(2):140, June 1955.

[Mor69] James B. Morris. E-resolution: Extension of resolution to include the equality relation. In Donald E. Walker and Lewis Norton, editors, *Proceedings 1^{st} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 287–294, Washington, D.C., USA, May 1969.

[Non86] Andreas Nonnengart. Vervollständigung von Termersetzungssystemen mit Hilfe einer Repräsentation durch gerichtete, azyklische Graphen. Diplomarbeit, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 6750 Kaiserslautern, Germany, June 1986.

[NSS59] Allen Newell, J. C. Shaw, and Herbert A. Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264. UNESCO, Paris, June 1959.

[Ohl87] Hans Jürgen Ohlbach. Link inheritance in abstract clause graphs. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 3(1):1–34, 1987.

[Ohl89] Hans Jürgen Ohlbach. Abstraction tree indexing for terms. In Hans-Jürgen Bürckert and Werner Nutt, editors, *UNIF'89 Extended Abstracts of the 3^{rd} International Workshop on Unification, SEKI-Report SR-89-17*, pages 131–136, Postfach 3049, 6750 Kaiserslautern, Germany, 1989. Fachbereich Informatik, Universität Kaiserslautern.

[OL80] Ross A. Overbeek and Ewing L. Lusk. Data structures and control architecture for implemention of theorem-proving programs. In Wolfgang Bibel and Robert Kowalski, editors, *Proceedings 5^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 87*, Les Arcs, France, July 1980. Springer-Verlag, Berlin, Germany.

[OS88] Hans Jürgen Ohlbach and Jörg H. Siekmann. Using automated reasoning techniques for deductive databases. SEKI-Report SR-88-06, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1988.

[OS89] Hans Jürgen Ohlbach and Jörg H. Siekmann. The Markgraf Karl Refutation Procedure. SEKI-Report SR-89-19, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1989.

[Ove75] Ross A. Overbeek. An implementation of hyper-resolution. *Comp. Maths. with Applications*, 1:201–214, June 1975.

[Pau85] E. Paul. On solving the equality problem in theories defined by Horn clauses. In Bruno Buchberger, editor, *Proceedings European Computer Algebra Conference, EUROCAL'85, volume 2, Lecture Notes in Computer Science (LNCS) 204*, pages 363–377, Linz, Austria, April 1985. Springer-Verlag, Berlin, Germany.

[Pel86] Francis Jeffry Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 2:191–216, 1986.

[Pet83] Gerald E. Peterson. A technique for establishing completeness results in theorem proving with equality. *SIAM (Society for Industrial and Applied Mathematics) Journal of Computing*, 12(1):82–100, February 1983.

[Prä85] Axel Präcklein. Ein Reduktionsmodul für einen automatischen Beweiser. Diplomarbeit, Institut für Informatik I, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe 1, Germany, März 1985.

[Prä90] Axel Präcklein. Solving equality reasoning problems with a connection graph theorem prover. SEKI-Report SR-90-07, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, April 1990.

[Prä92a] Axel Präcklein (editor). The HADES Manual. SEKI-Working Paper to appear, Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald 15, 6600 Saarbrücken 11, Germany, 1992.

[Prä92b] Axel Präcklein (editor). The MKRP-User Manual. SEKI-Working Paper SWP-92-03, Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald 15, 6600 Saarbrücken 11, Germany, 1992.

[PS89] Simon L. Peyton-Jones and J. Salkid. The spinless tagless g-machine. In *Proceedings 1989 ACM Conference on Functional Programming Languages and Computer Architecture*, 1989.

[Red85] Udday S. Reddy. Narrowing as the operational semantics of functional languages. In *1985 Symposium on Logic Programming*, pages 138–151, Boston, Massachusetts, USA, 1985. IEEE Computation Society Press, 1730 Massachusetts Avenue, Washington, D.C. 20036-1903, USA.

[Rém82] J. L. Rémy. Étude des systèmes de réécriture conditionelles et applications aux types abstraits algébriques. Thèse d'État, Nancy, France, 1982.

[Ric78] Michael M. Richter. *Logikkalküle*, volume 43 of *Leitfäden der angewandten Mathematik und Mechanik*. B. G. Teubner, Stuttgart, Germany, 1978.

[Ric83a] Monique Rice. The construction of a complete minimal set of contextual normal forms. In J. A. van Hulzen, editor, *Proceedings European Computer Algebra Conference, EUROCAL'83, Lecture Notes in Computer Science (LNCS) 162*, pages 255–266, London, England, March 1983. Springer-Verlag, Berlin, Germany.

[Ric83b] Elaine Rich. *Artificial Intelligence*. International Student Edition. McGraw-Hill Book Company, Auckland, USA, 1983.

[Ric91] Jörn Richts. Implementierung verschiedener Narrow-Strategien für einen automatischen Beweiser. Projektarbeit, Universität Kaiserslautern, Fachbereich Informatik, Postfach 3049, 6750 Kaiserslautern, Germany, July 1991.

[RKKL85] Pierre Rety, Claude Kirchner, Hélène Kirchner, and Pierre Lescanne. NARROWER: A new algorithm for unification and its application to logic programming. In Jean-Pierre Jouannaud, editor, *Proceedings 1^{st} Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 202*, pages 141–157, Dijon, France, May 1985. Centre de Recherche en Informatique de Nancy, Springer-Verlag, Berlin, Germany.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 211 East 43^{rd} Street, New York, 10017, USA*, 12(1):23–41, 1965.

[Rug91] Rosa Ruggeri. *Una logica sorteta con unione e disgiunzione di sorte*. PhD thesis, Università di Catania, 1991.

[Rus87] Michaël Rusinowitch. Démonstration automatique par des téchniques de réécriture. Thèse de Doctorat d'État en Mathématique, Nancy, France, 1987.

[Rus91] Michaël Rusinowitch. Theorem proving with resolution and superposition. *Journal of Symbolic Computation, Academic Press, Inc., London, England*, 11:21–49, June 1991. First published as report 87-R-128, CRIN, Nancy, France 1987. Also published in Proceedings International Conference on Fifth Generation Computer Systems, Tokyo, Japan 1988.

[RW69] George A. Robinson and Larry Wos. Paramodulation and theorem-proving in first-order theories with equality. *Machine Intelligence, Edinburgh University Press, 22 George Street, Edinburgh, Scotland*, 4:135–150, 1969.

[SA92] Rolf Socher-Ambrosius. A goal oriented strategy based on completion. Report MPI-I-92-206, Max-Planck-Institut für Informatik, Im Stadtwald, 6600 Saarbrücken, Germany, February 1992.

[Sch88] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*. PhD thesis, Universität Kaiserslautern, 1988. Also published as Lecture Notes in Artificial Intelligence (LNAI) 395, Springer-Verlag, Berlin, Germany, 1990.

[Sch89] Manfred Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation, Academic Press, Inc., London, England*, 8:51–99, 1989. Also published as SEKI-Report SR-87-16, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1987.

[She53] Yuting Shen. Paradox of the class of all grounded classes. *Journal of Symbolic Logic*, 18(2):114, June 1953.

[Sho76] Robert E. Shostak. Refutation graphs. *Artificial Intelligence, North Holland Publishing Company, Amsterdam, The Netherlands*, 7(1):51–64, 1976.

[Sib69] E. E. Sibert. A machine-oriented logic incorporating the equality axioms. *Machine Intelligence, Edinburgh University Press, 22 George Street, Edinburgh, Scotland*, 4:103–133, 1969.

[Sie75] Jörg H. Siekmann. Stringunification. Memo CSM-7of, Essex University, 1975.

[Sie89] Jörg H. Siekmann. Unification theory. *Journal of Symbolic Computation, Academic Press, Inc., London, England*, 7:207–274, 1989.

[Sie92] AG Siekmann. Ω-MKRP. SEKI-Report to appear, Fachbereich Informatik, Universität des Saarlandes, Im Stadtwald 15, 6600 Saarbrücken 11, Germany, 1992.

[SK90] Joachim Steinbach and Ulrich Kühler. Check your ordering – Termination proofs and open problems. SEKI-Report SR-90-25 (SFB), Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, December 1990.

[SL91a] Wayne Snyder and Christopher Lynch. Basic paramodulation. Report, Computer Science Department, Boston University, Boston, Massachusetts 02215, USA, 1991.

[SL91b] Wayne Snyder and Christopher Lynch. Goal directed strategies for paramodulation. In R. V. Book, editor, *Proceedings 4^{th} Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 488*, pages 150–161, Como, Italy, 1991. Springer-Verlag, Berlin, Germany.

[SNMG87] Gert Smolka, Werner Nutt, José Meseguer, and Joseph A. Goguen. Order-sorted equational computation. In H. Ait-Kaci and M. Nivat, editors, *Conference on Resolution of Equations in Algebraic Structures (CREAS)*, Lakaway, Texas, USA, 1987.

[Spe84] Volker Sperschneider. Logik. Script for Lectures at the Universität Karlsruhe, 1984.

[SS81] Jörg H. Siekmann and Peter Szabo. Universal unification and regular equational ACFM theories. In Ann Drinan, editor, *Proceedings 7^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 532–538, Vancouver, Canada, 1981. William Kaufmann, Inc., 95 First Street, Los Altos, California 94022, USA.

[Ste88] Rick L. Stevens. Challenge problems from nonassociative rings for theorem provers. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 730–734, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[Ste90] Guy L. Steele. *Common Lisp – The Language*. Digital Press, 12 Crosby Drive, Bedford, Massachusetts 01730, USA, 2^{nd} edition, 1990.

[Sti84] Mark E. Stickel. A case study of theorem proving by the Knuth-Bendix method discovering that $x^3 = x$ implies ring commutativity. In Robert E. Shostak, editor, *Proceedings 7^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 170*, pages 248–258, Napa, California, USA, May 1984.

[Sti85] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 1(4):333–357, 1985.

[Sti86] Mark E. Stickel. A Prolog technology theorem prover: Implementation by an extended Prolog compiler. In Jörg H. Siekmann, editor, *Proceedings 8^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 230*, pages 573–587, Oxford, UK, 1986. Springer-Verlag, Berlin, Germany.

[Sun87] Sun Microsystems Inc. The SPARC™ Architecture manual. Manual 800-1399-08, Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain view, California 94043, USA, 415-960-1300, 1987.

[SW80] Jörg H. Siekmann and Graham Wrightson. Paramodulated connection graphs. *Acta Informatica, Springer-Verlag, Berlin, Germany*, 13:67–86, 1980.

[Tep89] Michael Tepp. Kombinationsverfahren für Unifikationsalgorithmen. SEKI-Working Paper SWP-89-05, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1989.

[Wal82] Christoph Walther. The Markgraf Karl Refutation Procedure PLL – A first-order language for an automated theorem prover. Interner Bericht 35/82, Institut für Informatik I, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe, Germany, 1982.

[Wal84] Christoph Walther. Ein mehrsortiger Resolutionskalkül mit Paramodulation. Interner Bericht 23/84, Institut für Informatik I, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe, Germany, 1984.

[Wan88] T. C. Wang. Case studies of Z-Module reasoning: Proving benchmark theorems for ring theory. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 3, 1988.

[War77] D. H. D. Warren. *Applied Logic – Its Use and Implementation as Programming Tool.* PhD thesis, University of Edinburgh, 1977.

[War83] D. H. D. Warren. An abstract Prolog instruction set. Technical report, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025-3493, USA, 1983.

[Wei89] Christoph Weidenbach. A resolution calculus with dynamic sort structures and partial functions. SEKI-Report SR-89-23, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1989.

[Wei91] Christoph Weidenbach. A sorted logic using dynamic sorts. MPI-Report MPI-I-91-218, Max-Planck-Institut für Informatik, Im Stadtwald, 6600 Saarbrücken, Germany, December 1991.

[Wei93] Christoph Weidenbach. A superposition calculus with dynamic sort structures and partial functions. MPI-Report to appear, Max-Planck-Institut für Informatik, Im Stadtwald, 6600 Saarbrücken, Germany, 1993.

[WH89] Patrick Henry Winston and Berthold Klaus Paul Horn. *Lisp.* Addison Wesley Publishing Company, Reading, Massachusetts, USA, 1989.

[WM88] Larry Wos and William McCune. Challenge problems focusing on equality and combinatory logic: Evaluating automated theorem-proving programs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 714–729, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[WOH80] Larry Wos, Ross Overbeek, and Lawrence J. Henschen. Hyperparamodulation: A refinement of paramodulation. In Wolfang Bibel and Robert Kowalski, editors, *Proceedings 5^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 87*, pages 208–219, Les Arcs, France, July 1980. Springer-Verlag, Berlin, Germany.

[WOLB84] Larry Wos, Ross A. Overbeek, Ewing L. Lusk, and Jim Boyle. *Automated Reasoning Introduction and Applications.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1984.

[Wos88] Larry Wos. *33 Basic Research Problems.* Automated Reasoning. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1988.

[WRCS67] Larry Wos, George A. Robinson, Daniel F. Carson, and Leon Shalla. The concept of demodulation in theorem prooving. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 211 East 43rd Street, New York, 10017, USA*, 14(4):698–709, October 1967.

[YS86] Jia-Huai You and P. A. Subramanyou. E-unification algorithms for a class of confluent term rewriting systems. In Laurent Kott, editor, *Proceedings Automata, Languages and Programming, 13th Colloquium (ICALP), Lecture Notes in Computer Science (LNCS) 226*, pages 454–463, Rennes, France, July 1986. Springer-Verlag, Berlin, Germany.

[Zha88] Hantao Zhang. Reduction, superposition, and induction: Automated reasoning in an equational logic. Technical report 88-06, Department of Computer Science, The University of Iowa, Iowa City, Iowa 52242, USA, November 1988.

[ZK88] Hantao Zhang and Deepak Kapur. First order theorem proving using conditional rewrite rules. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9th International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 1–20, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[ZR85] Hantao Zhang and J. L. Rémy. Contextual rewriting. In Jean-Pierre Jouannaud, editor, *Proceedings 1st Conference on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science (LNCS) 202*, pages 46–62, Dijon, France, May 1985. Springer-Verlag, Berlin, Germany.

Index

abstract assembler 57
abstract code 58, 61
abstraction tree 65
AC-AC1 comparison 77
acyclic graph 65
as complete as 29
atom 6

B-G 45
Bachmair-Ganzinger 45
basic paramodulation 21
binding 56

C-G 45
classification 73
clause 6
clause graph calculus 7, 29
clause graph reduction 75
codomain 6
compilation 53, 54, 64
confluent 15
Connection-Graph 45
conservation of problem complexity 77
constant symbol 5
contextual rewriting 21
contraction rule 7
control strategy 39
counting strategy 49
critical pair construction 16
critical pair reduction 51

D 45
deduction rule 7
deduction rule scheme 7
Dershowitz 45
disagreement set 11
domain 6
dynamic sorts 75

E-refutation-graph 39
E-resolution 10
E-unification 8
E-resolution 27
equation chain 79
expansion rule 7

formula 6
function symbol 5

group 65

H-C 45
heuristic approach 41
heuristic level 42
Heuristic-Completion 45
Horn superposition 20
human proof 41
hyperparamodulant 24
hyperparamodulation 24

indexing 41, 53
indexing trees 64
induction 27, 74
infinitary theories 78
integration scratchpad 69
interpreter 66
iterative deepening 54

linear proof 40
link
    PE 41
    R 7
    R1 7
    R2 7
    S 7
    S1 7
    S2 7
link condition 8
link inheritance 7
Lisp programs 56
literal 6

Martelli-Monatanari 12
matcher 6
maximal 18
    strictly 18
merging paramodulation 19
merging superposition 20

mgu 8
most general unifier 6, 8, 77

narrowing 22, 26, 66
narrowing strategies 23
narrowing superposition 20
narrowing tree 26
natural deduction proof 79
Noetherian 15
NRF 11

ordered clause graphs 30
ordered paramodulation 18
ordering strategy 44
orderings 44

paramodulation 9
  basic 21
  hyper 24
  merging 19
  ordered 18
PE-link 41
position 6
predicate symbol 6
Prolog 53
proof transformation 79
PTTP 54
purity deletion 34

R-link 7
R1-link 7
R2-link 7
reduction ordering 47
reduction rule 7
reduction rule scheme 7
replacement 6
replacement paramodulation 36
replacement resolution 35
resolution 7
restriction strategy 44
rewrite 16
rewrite rule compilation 53
RRM 53
RUE 11

S-L 45
S-link 7
S1-link 7
S2-link 7
selection ordering 44
set-of-support weighting 50
single functions 60
smallest component strategy 49
Snyder-Lynch 45
sorts 75
splitting 75
steamroller 45
strategy reduction 72
strict clausal superposition 19
strict superposition 19
strictly maximal 18
structure sharing 53
structured axioms 74
substitution 6
subsumption 34
subterm 6
superposition
  Horn 20
  merging 20
  narrowing 20
  strict 19
  strict clausal 19
  weak clausal 18
superposition as strategy 28
superposition calculi 17

term 6
terminator situation 75
transformation 55, 57

unification 8, 12, 55
  using narrowing 23
unifier 6
unit superposition 20

variable symbol 6

WAM 53
weak clausal superposition 18

Z-K 45
Zhang-Kapur 45