Claude Kirchner (Ed.)

Rewriting Techniques and Applications

5th International Conference, RTA-93
Montreal, Canada, June 16-18, 1993
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editor

Claude Kirchner
INRIA Lorraine and CRIN
615 Rue du Jardin Botanique, F-54602 Villers les Nancy Cedex, France

CR Subject Classification (1991): D.3, F.3.2, F.4, I.1, I.2.2-3

ISBN 3-540-56868-9 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-56868-9 Springer-Verlag New York Berlin Heidelberg

Printed in Germany

Typesetting: Camera ready by author
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Preface

This volume contains the proceedings of RTA-93, the Fifth International Conference on Rewriting Techniques and Applications held June 16–18, 1993, in Montreal, Canada.

There were 91 submissions to RTA-93 authored by researchers from countries including Canada, France, Germany, Italy, India, Japan, the Netherlands, the People's Republic of China, Russia, Spain, United Kingdom, and the United States of America. Papers covered many topics: term rewriting; termination; graph rewriting; constraint solving; semantic unification, disunification and combination; higher-order logics and theorem proving, with several papers on distributed theorem proving, theorem proving with constraints, and completion.

Each submission was reviewed by at least three program committee members or their outside referees. All the members of the program committee met on February 1993 in Nancy and selected 29 papers and 6 system descriptions demonstrated during the conference and documented in this volume.

As for the proceedings of the previous conference, I welcomed the idea of presenting in the proceedings a list of open problems in the field and an update of the previous list of such open problems, showing altogether the strong activity of the term rewriting community in the large.

Three invited speakers gave a talk on their recent works related to the topics of RTA. Sergei Adian presented his work on algorithmic problems for groups and semigroups, Leo Bachmair the impact of rewriting techniques on theorem proving and Jean Gallier a general method for proving properties of typed lambda terms.

I am very grateful to the program committee for their efforts and cooperation in deciding the program and other related matters to RTA-93; to Mitsuhiro Okada for taking great care of the local arrangements for the conference; to the invited speakers Sergei Adian, Leo Bachmair and Jean Gallier, and lastly to Marian Vittek for doing everything that needed to be done to facilitate my task in organizing the program committee.

RTA-93 was sponsored by INRIA (France), the Centre de Recherche en Informatique de Nancy (France), Concordia University (Canada), the Center for Pattern Recognition and Machine Intelligence, Montreal (Canada), the Natural Science and Engineering Research Council (Canada), le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (Quebec) and the National Science Foundation (USA), and was held under the auspices of the European Association for Theoretical Computer Science.

Nancy, April 1993

Claude Kirchner
Chair, RTA-93

Program committee

Hubert Comon (Orsay)
Bruno Courcelle (Bordeaux)
Harald Ganzinger (Saarbrücken)
Jieh Hsiang (Stony Brook)
Claude Kirchner (Nancy)
Jan Willem Klop (Amsterdam)
Klaus Madlener (Kaiserslautern)
Paliath Narendran (Albany)
Mike O'Donnell (Chicago)
Mitsuhiro Okada (Montreal)
Leszek Pacholski (Wroclaw)
Michael Rusinowitch (Nancy)
Mark Stickel (Menlo Park)

Organizing committee

Ronald Book (Santa Barbara)
Nachum Dershowitz (Urbana)
Jean Gallier (Philadelphia)
Deepak Kapur (Albany)
Claude Kirchner (Nancy)
Klaus Madlener (Kaiserslautern)
Pierre Lescanne (Nancy)
David Plaisted (Chapel Hill)

Local arrangements

Mitsuhiro Okada (Montreal)

Referees

A. Arnold	P. Audebaud	J. Avenhaus
H.-J. Bürckert	L. Bachmair	S. Bailey
D. Basin	H. Baumeister	A. Bockmayr
F. de Boer	M.P. Bonacina	A.M. Borzyszkowski
A. Boudet	W. Bousdira	P. Casteran
W. Charatonik	T. Chen	C. Choffrut
E.A. Cichon	E. Contejean	T. Deiss
B. Delsart	J. Denzinger	E. Domenjoud
D. Dougherty	F. Fages	M. Falaschi
D. Fehrer	M. Fernández	M.C.F. Ferreira
R. Fettig	A. Geser	R. Gilleron
G. Gonthier	B. Gramlich	S. Hölldobler
M. Hanus	D. Hofbauer	J. Hong
M. Huber	U. Hustadt	P. Jacquet
P. Johann	J.-P. Jouannaud	T. Jurdziński
X. Kühler	R. Kennaway	D. Kesner
A. Kisielewicz	H.-J. Kreowski	D. Krob
M. Kutyłowski	S. Lange	D. Lugiez
C. Lynch	F. Müller	C. Marché
R. McCloskey	W. McCune	R. McNaughton
A. Middeldorp	B. Mu	R. Nieuwenhuis
T. Nipkow	D. Niwiński	V. van Oostrom
E. Orlowska	F. Otto	M. Parigot
M. Piotrów	D. Plaisted	E. Poll
L. Puel	D. Rémy	C.R. Ramakrishnan
P. Rao	S.A. Rebelsky	B. Reinert
P. Rety	M.M. Richter	C. Ringeissen
W. Sadfi	G. Salzer	A. Sattler-Klein
M. Schmidt-Schauss	K. Schulz	H. Seidl
D.J. Sherman	A. Skowron	G. Smolka
W. Snyder	R. Socher-Ambrosius	Z. Spławski
J. Steinbach	J.-M. Steyaert	J. Stuber
S. Tison	J. Tiuryn	X. Toenne
Y. Toyama	P. Urzyczyn	S. Vorobyov
U. Waldmann	I. Walukiewicz	R. Wichagen
C.P. Wirth	H. Zantema	H. Zhang

Table of Contents

Rewrite Techniques in Theorem Proving

Leo Bachmair
Department of Computer Science
University at Stony Brook
Stony Brook, New York, U.S.A.

The replacement of equals by equals is a common form of equational reasoning, of which rewriting is a refinement. Rewrite systems are sets of directed equations, called rewrite rules, that are used for replacements in the indicated direction only. A given expression is rewritten until a simplest possible form, a normal form, is obtained. Thus, the theory of rewriting is in essence a theory of normal forms. If a rewrite system is convergent, then all possible sequences of rewrites of equal terms result in the same normal form. In theories represented as convergent rewrite systems equality can therefore be decided rather efficiently.

Many aspects of the theory of rewriting can also be applied to resolution-style theorem provers. For instance, convergence requires that all sequences of rewrites terminate, a property that can be characterized by certain well-founded orderings called simplification orderings. If a total simplification ordering is imposed on a Herbrand base, then ground instances of a clause can be interpreted as conditional rewrite rules and refutational theorem proving may be viewed as a rewrite process: the negation of a theorem (the "goal") is rewritten until a contradiction is obtained. This method is only (refutationally) complete, though, if the set of rewrite rules extracted from the given clauses is convergent, and in general additional clauses may have to be deduced.

In this talk, I will discuss the fundamental techniques on which this rewrite approach to theorem proving is based. Two concepts are of particular interest: constraints and redundancy. Constraints provide a convenient way of describing the connection between the ground level (which embodies the interpretation of theorem proving as a rewrite process) and the general inferences that are actually applied by a prover to given clauses. In this context they have mainly been used to describe unification problems, ordering restrictions, and also certain normal-form properties of terms. Redundancy, on the other hand, allows one to optimize the proof search, as redundant formulas can be deleted and redundant inferences be ignored by a theorem prover.

Redundancy Criteria for Constrained Completion

Christopher Lynch Wayne Snyder*

April 2, 1993

Abstract

We study the problem of completion in the case of equations with constraints consisting of first-order formulae over equations, disequations, and an irreducibility predicate. We present several inference systems which show in a very precise way how to take advantage of redundancy notions in the context of constrained equational reasoning. A notable feature of these systems is the variety of tradeoffs they present for removing redundant instances of the equations involved in an inference. This combines in one consistent framework almost all practical critical pair criteria, including the notion of Basic Completion. In addition strict improvements of currently known criteria are developed.

1 Introduction

This paper presents a framework for exploiting redundancy notions in the context of a completion procedure for constrained equations. The constraint language consists of first-order formulae over atomic constraints consisting of equations, disequations, and an irreducibility predicate. An inference system is presented which shows precisely the tradeoffs involved in modifying constraints in order to delete unnecessary instances of the equations involved. The notion of redundancy we use is due to Bachmair and Ganzinger [1], and amounts to a semantic version of the well-known subconnectedness criterion (see [3]). Building on recent work on Basic Completion [4, 10], on constrained completion [8], and on various critical pair criteria [16, 13, 9] (see [3] for a survey), we show how a wide variety of techniques for removing redundant equations can be combined and refined in a consistent framework.

*Computer Science Department, Boston University, 111 Cummington St., Boston, MA 02215, U.S.A., lynch,snyder@cs.bu.edu.

Special cases of this inference system show how to implement a strict improvement of the technique of Basic Completion, and a stronger, hereditary version of the criterion based on subsumed critical pairs. In addition, we analyze the effect of initial constraints on the computation of critical pairs. It is hoped that this research contributes to the further development of the theory of constrained equational reasoning and to the practical improvement of existing completion procedures.

2 Preliminaries

We assume the reader is familiar with the standard definitions of terms constructed from a given set of symbols (augmented with an infinite set of Skolem constants). A *multiset* is an unordered collection with possible duplicate elements. An *equation* is a binary multiset $\{s,t\}$, conventionally represented $s \approx t$, where s and t are first-order terms over the given signature. A *substitution* is a mapping from variables to terms, e.g., $\{x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots\}$, the *domain* of a substitution σ as the set $Dom(\sigma) = \{\, x \mid x \neq x\sigma \,\}$. The application of a substitution σ to a term t is denoted $t\sigma$; if τ and ρ are substitutions, then $x\tau\rho = (x\tau)\rho$, for all variables x.

We assume that a reduction ordering $\succ$ (i.e., a well-founded ordering closed under substitution and context application) total on ground terms is given. Such an ordering can be extended to a well-founded ordering $\succ_{mul}$ on finite multisets of terms in the usual way. The ordering $\succ$ on equations is simply $\succ_{mul}$ restricted to binary multisets. The *maximum* of a set S of equations, denoted $max(S)$, is defined as the smallest $S' \subseteq S$ such that $\forall B \in S, \exists B' \in S', B \preceq B'$. We denote an equation $s \approx t$ where $s \succ t$ by an expression $s \to t$ and call it a *rewrite rule*; note in this case that we must have $Var(t) \subseteq Var(s)$.

The constraint language we shall use is a modification of the one presented in [8] to account for irreducibility constraints. For additional information on constraints, see [8] and references presented there.

Definition 1 *The set of constraints $\mathcal{C}$ is defined inductively as the smallest set of expressions containing the* atomic constraints $\top$, $\bot$, $s = t$, *and* $Irr(s)$ *(for every pair of terms s, t), and such that whenever φ_1 and φ_2 are in $\mathcal{C}$, then so are* $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \wedge \varphi_2)$, $\neg(\varphi_1)$, $(\exists x.\varphi_1)$, *and* $(\forall x.\varphi_1)$.

A constraint $\neg(s = t)$ is called a disequation.

The set of free variables in a constraint φ, denoted $Var(\varphi)$, is defined in the usual way. These are the variables that the constraint in fact constrains,

and solutions are substitutions over these variables. We typically use φ and ψ to denote constraints.

Definition 2 *Let R be a ground rewrite system. We define the solutions $Sol_R(\varphi)$ of a constraint φ relative to R inductively as follows. First, $Sol_R(\bot) = \emptyset$. Then, for any ground substitution σ,*

(i) $\sigma \in Sol_R(\top)$;

(ii) $\sigma \in Sol_R(s = t)$ iff $s\sigma = t\sigma$;

(iii) $\sigma \in Sol_R(Irr(s))$ iff $s\sigma$ is R-irreducible;

(iv) $\sigma \in Sol_R(\varphi_1 \wedge \varphi_2)$ iff $\sigma \in Sol_R(\varphi_1) \cap Sol_R(\varphi_2)$;

(v) $\sigma \in Sol_R(\varphi_1 \vee \varphi_2)$ iff $\sigma \in Sol_R(\varphi_1) \cup Sol_R(\varphi_2)$;

(vi) $\sigma \in Sol_R(\neg\varphi)$ iff $\sigma \notin Sol_R(\varphi)$;

(vii) $\sigma \in Sol_R(\exists x.\varphi)$ iff there exists some ground term t such that $\{x \mapsto t\}\sigma \in Sol(\varphi)$; and

(viii) $\sigma \in Sol_R(\forall x.\varphi)$ iff for every ground term t, $\{x \mapsto t\}\sigma \in Sol(\varphi)$.

Thus, each constraint and each ground rewriting system define a set of ground substitutions; a non-ground substitution σ is said to be a solution if every ground substitution $\sigma\tau$ is a solution. A constraint is *satisfiable* relative to R if there exists some solution; if no solution exists, it is *unsatisfiable*, and equivalent to $\bot$. We say that φ is *stronger than* or a *a strengthening of* ψ if for any R, $Sol_R(\varphi) \subseteq Sol_R(\psi)$; alternately, ψ is *weaker than* or *a weakening of* φ.

Note that this is *not* a set of solutions wrt a theory R, as in [8]; the rewrite system R is only used for the irreducibility constraints. An irreducibility constraint $Irr(s)$ can be used to forbid inferences into particular subterms of an equation which are known to be irreducible, for example if they are produced by application of a substitution; this is a particular kind of redundancy check, called the Basic Strategy in [4], which here is developed further in the context of equational and disequational constraints. In addition, we shall propagate irreducibility constraints through inferences. Irreducibility constraints in completion are used in the context of an evolving rewrite system which successively approximates the limit canonical system (this limit system is represented by R in the preceeding definition); thus in practice we can only state that a constraint $Irr(s)$ in the context of a current rewrite system R' is false when s is reducible by R'; in general we could never say that such a constraint is true until the limit system is reached. However, this will be sufficient to develop an extension to the Basic Strategy in our setting.

In the sequel an idempotent substitution could be considered to be a conjunction of equations; we shall make free use of this below, for example forming a new constraint by adding a substitution, e.g., $\varphi \wedge \sigma$.

A *constrained equation* is simply an equation between two terms plus a constraint, e.g., $s \approx t \,[\varphi]$. (Later we shall extend this notation to append other constraints to the equation.) The constraint determines which ground instances of the equation are available. Since an equation A without a constraint can be considered to be a constrained equation $A[\top]$, in the sequel we use the word *equation* in general to denote a constrained equation. The symbols A, B, etc. will be used to denote either an equation with its constraint or simply the equation part, depending on the context. The *erasure* of an equation $A[\varphi]$ is defined as $A[\top]$ and similarly for sets of equations. By $\varphi\sigma$ we denote the replacement of each free occurrence of $x \in Dom(\sigma)$ in φ by $x\sigma$. We assume the normal conventions for avoiding free variable capture. Any free variable in φ which does not occur in A is assumed in $A[\varphi]$ to be existentially quantified at the innermost possible level.

For any ground rewriting system R, the set of *ground instances* of an equation $A[\varphi]$ *relative to* R is defined as

$$Gr_R(A[\varphi]) \;=\; \{\, A\sigma \mid \sigma \text{ ground},\ Var(A) \subseteq Dom(\sigma),\ \text{and } \sigma \in Sol_R(\varphi)\}.$$

The set of ground instances of a set E is then defined

$$Gr_R(E) \;=\; \bigcup_{A \in E} Gr_R(A).$$

Remark In order to preserve completeness, we only allow a constraint of the form $s \approx t[\ldots Irr(u)\ldots]$ if either $u \prec s$ or $u \prec t$. If this restriction does not hold, then $[\ldots Irr(u)\ldots]$ is weakened to the form $[\ldots \bot \ldots]$ if $Irr(u)$ occurs *negatively* (i.e., in the scope of an odd number of negations). If the restriction does not hold and $Irr(u)$ occurs *positively*, for u is a constant or a variable, then $[\ldots Irr(u)\ldots]$ is weakened to the form $[\ldots \top \ldots]$; but if $u = f(u_1, \ldots, u_n)$, we can weaken the constraint into the form $[\ldots Irr(u_1) \wedge \ldots \wedge Irr(u_n)\ldots]$; this decomposition of the term must be iterated just until the restricted form is attained. We shall assume in the sequel that all equations have this restricted form.

3 Redundancy and Constraints

In this paper we present a strong inference system for constrained completion. We show the various tradeoffs which can be employed when applying redundancy notions [1] to eliminate certain instances of constrained equations involved in the inferences. Intuitively, a redundant equation is an

equation which is implied by smaller equations. Such equations are unnecessary in completing a set of equations. Our current formulation owes much to the paper [4].

Definition 3 *Let R be a ground rewriting system and E a set of equations. A ground instance $A \in Gr_R(E)$ is* R-redundant in E *if there exist equations $\{A_1, \ldots, A_n\} \subseteq Gr_R(E)$ such that $A_i \prec A$ for $1 \leq i \leq n$, and such that if each A_i is true in R, then A is true in R. If A is a non-ground equation, then A is R-redundant in E if every $A' \in Gr_R(A)$ is. If it is R-redundant, for any R, then it is simply called redundant.*

Now let $M = \{B_1, \ldots, B_k\}$ be a set of equations. We say that A is R-redundant in E upto M *if for each ground instance $A\sigma \in Gr_R(A)$, there exist equations $\{A_1, \ldots, A_n\} \subseteq Gr_R(E)$ and for each $B_j \in M$ there exists a ground instance $B'_j \in Gr_R(B_j\sigma)$ such that $A_i \prec max(B'_1, \ldots, B'_k)$ for $1 \leq i \leq n$, and such that if each A_i is true in R, then A is true in R.*[1]

For instance, equations with only identity instances are trivially redundant. In this paper we present a framework for representing redundancy information explicitly in an equation, by adding constraints to the equation which give more information about which instances are redundant; this information can then be propagated during inferences under certain conditions. Our notation uses an equation and a triple represented as $A[\![\varphi_1, \varphi_2, M]\!]$, where A is an equation, M is a set of equations, and φ_1 and φ_2 are constraints. We can think of this as an extension of the original notation $A[\![\varphi]\!]$, so that the first constraint φ_1 still represents the available instances of the equation, i.e., $Gr_R(A[\![\varphi_1, \varphi_2, M]\!]) = Gr_R(A[\![\varphi_1]\!])$. The other constraint and the set M record redundancy information in the following way.

Definition 4 *A constrained equation $A[\![\varphi_1, \varphi_2, M]\!] \in E$ is* correct for E *(or simply* correct *if E is obvious) if for all rewrite systems R (1)$Gr_R(A[\![\varphi_1]\!]) \subseteq Gr_R(A[\![\varphi_2]\!])$, (2)If $B \in Gr_R(A[\![\varphi_2]\!]) \setminus Gr_R(A[\![\varphi_1]\!])$ then B is R-redundant in E, and (3) If $B \in Gr_R(A) \setminus Gr_R(A[\![\varphi_2]\!])$ then B is R-redundant in E up to M.*

For example, an unconstrained equation has the form $A[\![\top, \top, \{A\}]\!]$. We will hereafter assume that all equations are in correct form, but may eliminate a suffix of the parameters if desired. If M is missing we assume it is $\{A\}$ and a missing φ_2 is assumed equal to φ_1, and a missing φ_1 is assumed to be $\top$. The last two components are used to store information about the history of an equation. Essentially, redundancy is used in the

[1] The point of this rather complex definition will be made clear in a moment.

completeness proof to show when equations become true. In passing such information around the inference system, it becomes useful to separate the ordering requirements in the definition of redundancy (e.g., "$A_i \prec A$") from the logical requirements (e.g., "if each A_i is true in R ..."). We thus wish to know when A is implied by equations smaller than B, and the set M preserves information about the smallest such B. It records which axioms S (original equations) were used to construct a given equation A. Clearly, S implies A, and thus (roughly) we let $M = max(S)$ in A. This parameter does not change for any particular equation.

We will use this redundancy information to delete instances of equations. For example, it is well known that overlaps at variable positions are not necessary. This is because instances with reducible substitutions are redundanct. In our framework we make this explicit, representing the irreducibility condition in the constraint. An unconstrained equation $fx \approx gx$ would be represented in correct form here as $fx \approx gx[Irr(x), \top]$. It is sufficient to to consider cases where the constraint is false to simulate the "no overlaps at variable positions" condition and also the Basic strategy.

4 Constrained Critical Pair Generation

In this section we give a generalization of the critical pair rule from [8] and show how a variety of tradeoffs may be obtained in deleting various instances of the equations involved in an inference.

The general form of our constrained critical pair rule is

C-Deduce

$$\frac{s \to t[\varphi_1, \varphi_2, M] \qquad u[s'] \to v[\psi_1, \psi_2, N]}{u[t]\sigma \approx v\sigma[\Delta_1, \Delta_2, max(M\sigma \cup N\sigma)]}$$

where (1) $\sigma = mgu(s, s')$, (2) Δ_1 is a weakening of $\varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(s\sigma)$, (3) Δ_1 is a strengthening of $Irr(x_1) \wedge \ldots \wedge Irr(x_n)$, where $\{x_1, \ldots, x_n\} = Var(u[t]\sigma \approx v\sigma)$, (4) the conclusion is a correct equation, and (5) after constucting the conclusion we may potentially modify some of the premise constraints as long as these are still correct equations.

In general in the inference rules we present, equations will have the form $A[Irr(s_1) \wedge \ldots \wedge Irr(s_n) \wedge \varphi_1', \varphi_2, M]$, where any variable in A occurs in some s_i. Note that we have not explicitly stated the condition "where s' is not a variable," but in fact this will be a consequence of the irreducibility constraints built up during the inference process. Inferences involving variable overlaps can be shown to be redundant and hence unnecessary.

The correctness criteria here basically assert that if instances of these equations are deleted by the inference, then these instances are redundant.

The general idea of the various instances of this schema we present is that certain instances of the right premise are redundant by virtue of certain instances of the left premise and the conclusion; the tradeoffs occur in considering whether we want to strengthen the right premise by deleting as many instances of the right premise as possible, in which case we need perhaps to weaken the other equations by making more instances available, or whether we wish to strengthen the conclusion as much as possible, in which case we can not delete as many instances of the right premise. Essentially these rules can be thought of as combinations of simplification and overlap rules. In addition, it is possible to define situations under which the inference itself is redundant and hence need not be performed.

Definition 5 *For any R and E, a C-Deduce inference as given above is R-*redundant *in E if (i) the σ-instance of either premise is R-redundant in E, or (ii) $u[t]\sigma \approx v\sigma[\varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(s\sigma), \Delta_2, max(M\sigma \cup N\sigma)]$ is R-redundant in E. If it is R-redundant, for any R, then it is simply called redundant.*

To present these inference rules we need to say what the values of the constraints in the conclusion are, and how the constraints in the premises are (potentially) modified. For each case, we would need to show that the conditions of C-Deduce are satisfied; we omit these proofs from this abstract. First we present two general constraint modification rules that may be applied to strengthen the right premise after an inference has been performed.

Let **CM1** be the right premise constraint modification rule: $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge \Delta_2 \wedge \varphi_2)$, and let **CM2** be the rule: $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge \Delta_2)$.

It can be shown that if $s\sigma \rightarrow t\sigma \prec u[s']\sigma \rightarrow v\sigma$ then CM1 applied to the right premise yields a new correct equation.[2] If in addition we have $M\sigma \prec_{mul} \{u[s']\sigma \rightarrow v\sigma\}$ then CM2 applied to the right premise yields a correct equation.

The first inference system presented is called CCP (Constrained Critical Pairs). In this case the conclusion is as strong as possible, the left premise is not weakened, and some instances of the right premise are deleted.

Definition 6 *Let CCP be the instance of C-Deduce where $\Delta_1 = \Delta_2 = \varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(s\sigma)$, and where CM1 is performed if $s\sigma \rightarrow t\sigma \prec u[s']\sigma \rightarrow v\sigma$.*

In the CCP inference Δ_1 is as strong as it can be in an inference. Given the value of Δ_1 we could try to make Δ_2 as weak as possible so we can delete more of the instances of the right premise. For example, if the conclusion

[2] Note that if this condition is violated then the conclusion is either unorientable or an identity.

is $fx \approx gx[Irr(x), Irr(x)]$, then we could change Δ_2 to $\top$, because all reducible instances are redundant. In general, if $\Delta_1 = Irr(x) \wedge \varphi'$ and φ' does not further constrain x, then Δ_2 can be set equal to φ' (this process can be iterated). Call the result of this iteration $NoIrrVar(\Delta_1)$. Although we shall have occasion to refer to this notion in a later section, for simplicity in this abstract, we have presented a simpler version where $\Delta_1 = \Delta_2$.

Our second instance of C-Deduce emphasizes strengthening the right premise as much as possible, essentially by simplifying as many instances of the right premise as possible by instances of the left premise. In this case we may have to weaken the left premise and construct a weaker conclusion than in the previous rule.

Definition 7 *C-Simplify is the instance of C-Deduce such that* $\Delta_1 = \Delta_2 = \psi_1\sigma$, *and where in addition if* $s\sigma \rightarrow t\sigma \prec u[s']\sigma \rightarrow v\sigma$ *we change* ψ_1 *in the right premise to* $\psi_1 \wedge \neg\sigma$; *finally, unless* $M\sigma \prec_{mul} \{u[s']\sigma \rightarrow v\sigma\}$ *holds we must further modify the premise constraints so that* $\varphi_1 \Rightarrow \varphi_1 \vee (\sigma \wedge \psi_1 \wedge \neg\varphi_2)$ *and* $\varphi_2 \Rightarrow \varphi_2 \vee (\sigma \wedge \psi_1)$.

These two rules illustrate the range of tradeoffs available. In CCP we do not weaken the conclusion or the left premise, so that we can only eliminate some instances of the right premise. In C-Simplify we must weaken the constraints on the conclusion and the left premise in general but we can then delete all possible instances of the right premise. It is possible to define inference rules between these two extremes. In the next definition we present two rules which weaken the conclusion but not the left premise of the inference.

Definition 8 *Suppose* $s\sigma \rightarrow t\sigma \prec u[s']\sigma \rightarrow v\sigma$. *Then we define the rule CCP1 as the instance of C-Deduce where* $\Delta_1 = \Delta_2 = \varphi_2\sigma \wedge \psi_1\sigma \wedge Irr(s\sigma)$ *and with the strengthening* $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge \varphi_2 \wedge Irr(s\sigma))$. *If in addition, we have* $M\sigma \prec_{mul} \{u[s']\sigma \rightarrow v\sigma\}$, *then we may define the instance CCP2 of C-Deduce where* $\Delta_1 = \Delta_2 = \psi_1\sigma \wedge Irr(s\sigma)$ *and such that* $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge Irr(s\sigma))$.

In a similar manner it is possible to define other inference rules that partially weaken the conclusion and the left premise so some instances of the right premise are deleted. For instance we can weaken the constraints on the conclusion so that just the irreducibility constraints remain, or we can weaken the constraints so that just the equational and disequational constraints remain.[3] Thus it is possible to define a spectrum of possible critical pair rules in our framework.

[3] To be precise we would also need to keep the irreducibility constraints on the variables of the conclusion to avoid superposing into variables.

Now we consider some examples of the above inference rules. Consider the inference

$$\frac{fa \to b \qquad fx \to gx[Irr(x), \top]}{b \approx ga[Irr(a), Irr(a), fa \to ga]}$$

on axioms. If we use the CCP rule then we may apply CM1 to the constraint of the right premise: $Irr(x) \Rightarrow Irr(x) \wedge (x \neq a \vee \neg Irr(a))$. If we use C-Simplify then the conclusion becomes $b \approx ga[\top]$ and the right premise is modified by $Irr(x) \Rightarrow Irr(x) \wedge x \neq a$. We can now show how these two inferences would provide additional information usable in later inferences. Assume we followed the C-Simplify inference just given with

$$\frac{fx \to gx[Irr(x) \wedge x \neq a, \top] \qquad gfa \to c}{gga \approx c[\bot, \bot, gfa \to c]}$$

The first thing to note is that this inference is redundant because the constraint on the conclusion is unsatisfiable. Therefore the inference does not need to be performed. However, we may be interested in simplifying the right premise, so we still perform the inference. Using C-Simplify we get $gga \approx c[\top]$ for the conclusion. The first constraint on the right premise becomes $\bot$ which means that none of the instances of the equation are necessary. However, the second constraint is still $\top$ which means that all the instances are redundant. Therefore we may use it to simplify an equation if we like, without weakening the constraint, but we are never required to use it in an inference. This illustrates the benefit of the second constraint. If we had not saved the second constraint we would have had to weaken the first constraint on the left premise.

To illustrate the benefit of the third component of the constraint triple we consider following the CCP inference in the first example with

$$\frac{ga \to b[Irr(a), Irr(a), fa \to ga] \qquad fga \to ga}{fb \approx ga[Irr(a), Irr(a), fga \to ga]}$$

If we want this to be a C-Simplify inference the conclusion can be weakened to $fb \approx ga[\top, \top, fga \to ga]$. Then we can use CM2 to set the first constraint of the right premise to $\bot$ as in the previous example, since all instances of left premise are true by equations smaller than the right premise.

We give one more example to illustrate a use of the irreducibility constraints. Consider the inference

$$\frac{fa \to b \qquad fa \to ga}{b \approx ga[Irr(a), Irr(a), fa \to ga]}$$

We could consider this to be a C-Simplify inference, weaken the constraint in the conclusion and change the contraint of the right premise to $\bot$. If we used

CCP instead the first constraint on the right premise becomes $\neg Irr(a)$ using CM1. Any inference using this equation as left premise is now redundant because a must be irreducible in an inference. That is, when $fa \to ga$ is used as a left premise, fa can be restricted to be in normal form (cf. the prime superposition criterion discussed below), which violates the constraint.

Naturally, other rules for simplifying rhs's of rules, orienting, etc. are necessary for a practical system, but for brevity we have presented only our critical pair rule. These are relatively straightforward adaptations of the ideas above, except for the blocking rules, and are presented in full in the long version. Irreducibility constraints give us blocking rules based on the reducibility of terms in constraints $Irr(s)$. For example, suppose we have equations $A[...Irr(u[s'])...]$ and $s \to t[\varphi]$, where $s\rho = s'$. Then the first equation can be changed to $A[...(Irr(u[s']) \wedge \neg(\varphi\rho))...]$. Clearly if all instances of $s \to t$ are available, i.e., $\varphi = \top$, then this corresponds to solving the constraint $Irr(u)$ by replacing it with $\bot$.

In the remainder of this section we show how we can set the parameters of the C-Deduce rule to give other critical pair criteria as special cases of ours. To start with we consider standard completion.

The *standard critical pair rule* can be represented in our system by letting $\Delta_1 = Irr(x_1) \wedge \ldots \wedge Irr(x_n)$, where $\{x_1, \ldots, x_n\} = Var(u[t]\sigma \approx v\sigma)$, $\Delta_2 = \top$, and P be anything that yields a correct equation (since it will never be used). This is only necessary to disallow superposition into variable positions. The simplification rule can be represented by the same conclusion, with the right premise modified using CM1. Since simplification is only performed when σ is a matcher, the first constraint on the right premise becomes $\bot$ so the equation may be deleted.

Prime superposition [7] is a critical pair criterion which states that an inference is unnecessary if the lhs of the left premise is reducible. This follows directly from our redundancy criteria. An inference is redundant if $Irr(s\sigma)$ is unsatisfiable. In fact our results provide for a hereditary version of this criterion.

General superposition [16] and the critical pair criteria discussed in [9, 13, 14] are all examples of a more general principle of *subsumed critical pairs* [3]. Once an overlap on an equation A is produced, involving an *mgu* σ, then it is no longer necessary to consider overlaps on A involving *mgus* less general or equal to σ. We simulate these critical pair criteria with disequational constraints. The constraints on the conclusion would be the same as the constraints in the standard critical pair rule. The difference is that CM1 is then performed. The first constraint of the right premise then becomes $\psi_1 \wedge \neg\sigma$. This disallows further superpositions into the right premise where the *mgu* is less general than or equal to σ, since these in-

stances are no longer present. Again, our results provide for a hereditary version of this criterion. In other words, if a right premise has been overlapped with *mgu* σ, then the conclusion also never needs to be overlapped with an *mgu* less general or equal to σ.

In addition to naturally simulating subsumed critical pair criteria with our inference system we also naturally simulate *basic completion* [4, 10]. In this strategy, overlaps are disallowed on terms introduced by substitution. This is simulated with irreducibility constraints. In the conclusion of an inference we let $\Delta_1 = \varphi_1\sigma \wedge \psi_1\sigma$ and $\Delta_2 = NoIrrVar(\Delta_1)$. Essentially, all constraints would be conjunctions of irreducibility constraints; constraints on the variables of the premises are instantiated by the *mgu* which restricts us from superposing into those positions. In fact, we can obtain a stronger version, because constraints can be kept on terms not occurring in the equation. The special form of simplification required in basic completion can be simulated by our techniques for weakening the left premise.

The completion system in [8] is designed for a set of equations with initial constraints. The authors are not concerned with efficiency constraints and redundancy. As we have shown in the beginning of this section, completion is not complete with initially constrained equations unless we allow superposing into variables. In order to insure completeness [8] considered some additional inference rules which basically had the purpose of turning constrained equations into unconstrained equations. In our full paper in preparation we show how completeness can be preserved with initial constraints by allowing a limited form of variable ovelap. Our completeness proof is the first one we are aware of for equation and disequation constraints without any additional rules. We studied the combination of irreducibility constraints (to embed Basic Completion) with a subset of the constraints considered in [8]. For example we do not consider ordering constraints (see also [12] and [10]), although it seems they could be added to our system without major alterations of the framework.

We now consider the completeness of the rules presented in the previous section. For lack of space we can present no formal proofs, referring the reader to the full paper. We emphasize that we are considering only the critical pair rules here, and not the full complement of completion inference rules. It is sufficient for completeness however to consider only the critical pair rules.

Following the paradigm developed at length in the book [3], we define a *derivation* to model the process of completion.

Definition 9 *A sequence* $< S_0, S_1, \ldots >$ *of sets of equations is a* derivation *from* S *if* $S_0 = S$ *and for each* $i \geq 0$, *for any* R, $Gr_R(S_{i+1}) = (Gr_R(S_i) \cup E_1) \setminus E_2$ *where* E_1 *and* E_2 *are sets of equations such that* $Gr_R(S_i) \models E_1$ *and*

each equation in E_2 is R-redundant in $Gr_R(S_i) \cup E_1$. Let $S_\infty = \bigcup_j \bigcap_{k \geq j} S_k$. We call S_∞ the limit *of the derivation. Any equation $A \in S_\infty$ is called* persisting.

Definition 10 *Let I be some instance of C-Deduce. A derivation is an I-*derivation *if each S_{i+1} is obtained from S_i by application of the rule I. A set S is* I-saturated *if every I-inference from S is redundant. An I-derivation is* fair *if the limit is I-saturated.*

An inference rule can be viewed as a method for adding consequences to the set and deleting redundant instances. The next result shows that this is correct in the limit.

Lemma 1 *Let R be a ground rewriting system and suppose for two sets of equations E and E', $Gr_R(E) \subseteq Gr_R(E')$. (1) Any closure (or inference) which is R-redundant in E is also R-redundant in E'. (2) If all ground instances in $Gr_R(E') \setminus Gr_R(E)$ are R-redundant in E', then any equation (or inference) which is R-redundant in E' is also R-redundant in E.*

This shows that the inference systems presented are sufficient to saturate a set of equations. We now show that saturated sets are ground canonical. In our framework, this will allow us to argue that our constrained completion systems (which are not defined as unfailing) will produce canonical sets in the limit. Our proof follows very much in the lines of the proof in the journal version of [4], with the addition of the constraint formalism. In addition, there are some delicate features of the proof which relate to the use of the irreducibility constraints defined relative to a rewrite system which is constructed from the set of constrained equations itself. First we give a method for constructing a canonical set of ground rewrite rules from a given set of equations.

Definition 11 *Let E be a set of equations and $\mathcal{EQ}$ denote the set of all ground equations. We define the ground rewriting system R_E using induction on $(\mathcal{EQ}, \succ)$ by associating with each $A \in \mathcal{EQ}$ a rewrite system R_A. Assume for a ground equation A that R_B has been defined for each ground equation B with $B \prec A$, and let $R_{\prec A}$ be defined as $\bigcup_{B \prec A} R_B$. Then $R_A = \{A\}$ if A is a member of $Gr_{R_{\prec A}}(E)$ in the form $s \to t$ and where s irreducible by $R_{\prec A}$; otherwise $R_A = \emptyset$. Finally define R_E as $\bigcup_{A \in \mathcal{EQ}} R_A$.*

Notice that the rewrite system R_E is constructed out of instances from substitutions reduced relative to smaller rewrite rules already in R_E.

Let us say that a ground instance $A\sigma$ of an equation from E is *reduced relative to R*, or an *R-reduced instance of E*, if $x\sigma$ is irreducible by R for every $x \in Dom(\sigma)$. The properties of the preceding definition we shall need are as follows.

Lemma 2 *For R_E as just defined, (i) Every equation in R_E is an R_E-reduced instance of E; (ii) R_E and $R_{\prec A}$ for any $A \in \mathcal{EQ}$ are canonical; (iii) No equation $A \in R_E$ is true in $R_{\prec A}$; and (iv) An equation $A \in Gr_{R_E}(E)$ is true in R_E iff it is true in $R_{\prec A} \cup R_A$.*

Theorem 1 *Let I be some instance of the C-Deduce rule, R_E be as above, and E be an I- saturated set of equations such that for each $A[\varphi_1, \varphi_2, M] \in E$, φ_1 is stronger than $Irr(x_1) \wedge \ldots \wedge Irr(x_n)$ for $\{x_1, \ldots, x_n\} = Var(A)$. Then R_E makes true every member of $Gr_{R_E}(E)$.*

We now state the main completeness result of the paper.

Theorem 2 *Let E be a set of unconstrained equations and S be the set of equations $A[Irr(x_1) \wedge \ldots \wedge Irr(x_n), \top]$, for $A \in E$ and $\{x_1, \ldots, x_n\} = Var(A)$. Let $< S, \ldots >$ be an I-fair derivation from S for some instance I of C-Deduce. If S_∞ contains no unorientable equations then it is ground canonical and equivalent to E. In addition the erasure of S_∞ is a canonical rewriting system equivalent to E.*

The proof that the erasure is a canonical (and not just a ground canonical) rewrite system involves a Skolemization step, and is from [4]. This shows that our inference system (which was not presented as an unfailing completion procedure) produces a canonical rewriting system in the limit. If a derivation is finite, then of course the final system is canonical. In this case it could be considered to be a constrained rewriting system, or its erasure could be produced. The adaptation of these results to the case of unfailing completion is straightforward and left to the full paper.

5 Conclusion

We have presented several inference systems which show in a very precise way how to take advantage of redundancy notions in the context of constrained equational reasoning. These systems illustrate the tradeoffs involved in this framework in a very precise way. We hope that this research contributes to the further development of the theory of constrained equational reasoning and to the practical improvement of existing completion procedures.

The method of proof used in this paper was adapted from our previous paper with Bachmair and Ganzinger on Basic Paramodulation [4] (see also [10]), which in turn adapted the results of [1] (cf. [11] and [17]). However, the inference systems are developments of the rules from the seminal paper [8] to show how irreducibility constraints can be used to express the idea of Basic Completion in combination with other kinds of equational constraints.

To apply the procedures given in this paper, one needs to have a constraint solving algorithm. Comon and Lescanne [5] have analyzed the problem of solving constraints of equations and disequations. In our framework we also consider irreducibility constraints, however, which complicates the situation. For an arbitrary canonical system R, reducibility and irreducibility tests can be made using inductive reducibility and narrowing tests, however in our setting these tests must be made with respect to an evolving rewrite system and in the presence of constrained rewrite rules. Thus only certain tests can be made. Some of these have been explained in our blocking rules. In general, for an incompletely specified rewrite system, we can only know that if a term t is reducible at some stage, it will be reducible in the limit as well; we can never state in the positive that t is irreducible before the completion process terminates.

We do not expect that this framework in its entirety would be necessarily be an efficient and useable form of completion procedure. We instead view it as a theoretical model for constrained completion, some of whose special cases may turn out to be practically useful. Our current research focusses on simple and efficient subcases of the general framework which promise to eliminate as many redundant inferences and equations as possible without excess amounts of overhead. A particular focus is on subclasses for which efficient constraint solving techniques exist. The implementation of this system, and the Basic Completion system discussed in [4], is currently being investigated at BU as part of the Masters Thesis [6].

References

[1] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplifica tion. To appear in *Journal of Logic and Computation* (1992).

[2] L. Bachmair and N. Dershowitz. Critical Pair Criteria for Completion. *J. Symbolic Computation 6* (1988) pp.1-18.

[3] L. Bachmair. *Canonical Equational Proofs*. Birkhauser Boston, Inc., Boston MA (1991).

[4] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation and Superposition. In *Proc. 11th Conference on Automated Deduction*, Saratoga Springs, NY (1992) pp. 263–476. Journal version in preparation.

[5] H. Comon and P. Lescanne. Equational Problems and Disunification. *Journal of Symbolic Computation* 7 (1989) pp. 371-426.

[6] D. Durand, *Experiments in Basic Paramodulation and Constrained Completion*, Masters Thesis, Boston University Computer Science Department (1992).