Redundancy Criteria for Constrained Completion

Christopher Lynch Wayne Snyder*

March 5, 1995

Abstract

This paper studies 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 this framework. A notable feature of these systems is the variety of tradeoffs they present for removing redundant instances of the equations involved in an inference. The irreducibility predicates simulate redundancy criteria based on reducibility (such as prime superposition and Blocking in Basic Completion) and the disequality predicates simulate the notion of subsumed critical pairs; in addition, since constraints are passed along with equations, we can perform hereditary versions of all these redundancy checks. This combines in one consistent framework stronger versions of all practical critical pair criteria. We also provide a rigorous analysis of the problem with completing sets of equations with initial constraints. Finally, an interesting consequence concerning the recalculation of critical pairs in completion procedures is discussed.

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 schema 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] (see also [17]), and amounts to a semantic version of the well-known subconnectedness criterion. Building on recent work on Basic Paramodulation [4, 5, 15], on constrained completion [11, 15, 16], and on various critical pair criteria (see [3] for a survey), we show how a wide variety of techniques for removing redundant equations can be combined and strengthened in a consistent framework. The tradeoffs we explore are essentially refinements of the techniques for weakening constraints in the deletion rules of Basic Paramodulation. Special cases of this inference system show how to implement strict improvements of critical pair criteria based on reducibility (such as prime superposition and Blocking in Basic Completion) and on the notion of subsumed critical pairs. In addition, we analyze the effect of initial constraints on this system.[1] 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.

The paper is organized as follows. Following a section of preliminary definitions, in section 3 we formally discuss the relationship between constraints and redundancy, and then in section 4 present our inference system and discuss its general features and its relationship to other critical

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

[1] For a discussion of the effect of initial ordering constraints see [16].

pair criteria. In section 5 we discuss constraint solving with irreducibility constraints. In section 6 we develop the theoretical justification for this system and rigorously prove completeness. In section 7, we show how it is possible to apply our completion process without recalculating critical pairs after the left hand side of a rule is simplified. In section 8 we show how this framework can be used to analyze completion of sets of equations with initial constraints. We conclude with a comparison with previous work and a discussion of current and future directions.

2 Preliminaries

We present here a brief overview of the notation and preliminary definitions necessary for the paper; for a more thorough coverage, see the books [3, 20] and, in particular, the seminal paper [11].

2.1 Equations, Orderings, and Constraints

We assume the reader is familiar with the construction of a set of terms $T(\Sigma, X)$ from a given signature Σ of constant and function symbols and a set of variables X. In this paper we implicitly assume a fixed signature Σ and use Σ^+ to denote an arbitrary extension of this signature.

The set of variables occurring in a term t is denoted $Var(t)$. We generally use the letters l, r, s, t, u, and v for terms, and the letters w, x, y, and z for variables. A *multiset* is an unordered collection with possible duplicate elements. For any multiset M, the number of occurrences of an object x is denoted $M(x)$. 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. We often denote equations by the Roman letters A, B, C, and D. Sets of equations are denoted by E or R.

By a *ground* term or equation we mean one containing no variables. The set of ground terms constructed from a signature Σ is denoted $T(\Sigma)$. A *substitution* is a mapping from variables to terms, e.g., $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, which is almost everywhere equal to the identity, and is typically denoted by Greek letters σ, θ, η, ρ, or τ. We define 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, denoted $t\sigma$, is defined as usual. Composition of substitutions is denoted by juxtaposition; if τ and ρ are substitutions, then $x\tau\rho = (x\tau)\rho$, for all variables x. A substitution σ is said to be a *grounding substitution* for an object Γ if $\Gamma\sigma$ has no free variables.

We assume that a reduction ordering $\succ$ (i.e., a well-founded ordering closed under substitution and context application) total on $T(\Sigma)$ is given, and that this ordering can be extended to a reduction ordering total on any $T(\Sigma^+)$.[2]

Such an ordering can be extended to a well-founded ordering $\succ_{mul}$ on finite multisets of terms as follows: $M \succ_{mul} N$ if (i) $M \neq N$ and (ii) whenever $N(s) > M(s)$ then $M(t) > N(t)$, for some t such that $t \succ s$. The ordering $\succ$ on equations is simply $\succ_{mul}$ restricted to binary multisets. If E is a set of equations and A is an equation, we define $E_A = \{B \in E \mid B \prec A\}$, and $E_{A^+} = \{B \in E \mid B \preceq A\}$. 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 the expression $s \to t$ and call it a *rewrite rule*; note in this case that we must have $Var(t) \subseteq Var(s)$. An *orientable instance* of an equation $s \approx t$ is an instance $s\sigma \approx t\sigma$ such that $s\sigma \succ t\sigma$, or an instance $t\sigma \approx s\sigma$ such that $t\sigma \succ s\sigma$.

If S is a set of equations and E is true in every model of S we write $S \models E$. If S and T are two sets of equations such that, for all equations A, $S \models A$ if and only if $T \models A$, then we write $S \equiv T$ and say that S is logically equivalent to T. It is assumed that the reader is familiar with the standard notions of rewriting. A rewrite system which is terminating and confluent on ground

[2]For example, we can combine a well-founded precedence over Σ and a well-founded ordering of $\Sigma^+ \setminus \Sigma$ using ordinal addition to obtain a well-founded ordering on Σ^+ and then use the *lpo*.

terms is called *ground canonical*, and if it is terminating and confluent on all terms it is simply called *canonical*.

We also assume an ordering $<_r$ on all the subterms of the left hand side of each rewrite rule (where "r" stands for "redex ordering," as in [5]). The ordering must have the property that $s <_r t$ implies that t is not a subterm of s. Note that this ordering may be different for each equation, even if two equations have the same left hand side. Examples of such orderings are a postorder traversal, or a reduction ordering.

The constraint language we shall use is a modification of the one presented in [11] to account for 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, 5], which here is developed further in the context of completion with equational and disequational constraints. Other kinds of redundancy checks, such as Prime Superposition [10], also involve reasoning about reducibility of terms involved in an inference, and can be represented by irreducibility constraints. In addition, we shall propagate irreducibility constraints through inferences, thus providing hereditary versions of these redundancy checks, and also use them to explain the "no variable-overlaps" condition, and the modifications necessary to this condition when the initial set of equations contains constraints.

Irreducibility constraints in completion are used in the context of an evolving rewrite system which successively approximates the limit canonical system; 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. Note that if a term is reducible at some stage of the completion process by a rewrite rule, then it will be reducible at all later stages; thus once an irreducibility predicate becomes false it remains false. However, the only way we can say that such a constraint is true is when we can be assured that a term is in normal form in the limit, which is generally impossible at a finite stage of the completion process. But the partial information we have about reducibility at each finite stage of the process will be sufficient to eliminate many redundant equations and inferences, including those covered by all current critical pair criteria which use some notion of reducibility.

We now formally develop the notion of constraints we shall use. For additional information on constraints, see [11] 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 literal *is an atomic constraint or its negation. A literal $\neg(s = t)$ is called a* disequation, *abbreviated by $s \neq t$.*

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. Thus, constraints act as filters for the allowable instances of equations. We typically use φ and ψ to denote constraints. We observe the normal conventions for removing parentheses. In the sequel an idempotent substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ can be equivalently considered to be a constraint of the form $x_1 = t_1 \wedge \cdots \wedge x_n = t_n$. We shall make free use of this below, for example forming a new constraint by adding a substitution, e.g., $\varphi \wedge \sigma$. In addition, we extend the predicate $Irr()$ to *sets* of terms, where $Irr(\{t_1, \ldots, t_n\})$ is defined to be $Irr(t_1) \wedge \ldots \wedge Irr(t_n)$.

In what follows we have occasion to refer to a ground rewrite system R constructed from instances of equations. This system is only used to give a meaning to the predicate $Irr()$ in the proofs and is not part of the completion process, and hence is considered to be an *unconstrained* set of ground equations. We now define the meaning of a constraint relative to R (cf. [11]).

Definition 2 *Let R be a ground set of unconstrained equations over a signature Σ and contained in $\succ$. We define the solutions $Sol_R^\Sigma(\varphi)$ of a constraint φ relative to R inductively as follows. First, $Sol_R^\Sigma(\bot) = \emptyset$. Then, for any ground substitution σ with range $T(\Sigma)$,*

1. $\sigma \in Sol_R^\Sigma(\top)$;
2. $\sigma \in Sol_R^\Sigma(s = t)$ *iff* $s\sigma = t\sigma$;
3. $\sigma \in Sol_R^\Sigma(Irr(s))$ *iff* $s\sigma$ *is ground and there exists no* $s' \in T(\Sigma)$ *such that* $s' \prec s\sigma$ *and* s' *is equivalent to* $s\sigma$ *modulo* $R_{s\sigma \approx s\sigma}$;[3]
4. $\sigma \in Sol_R^\Sigma(\varphi_1 \wedge \varphi_2)$ *iff* $\sigma \in Sol_R^\Sigma(\varphi_1) \cap Sol_R^\Sigma(\varphi_2)$;
5. $\sigma \in Sol_R^\Sigma(\varphi_1 \vee \varphi_2)$ *iff* $\sigma \in Sol_R^\Sigma(\varphi_1) \cup Sol_R^\Sigma(\varphi_2)$;
6. $\sigma \in Sol_R^\Sigma(\neg\varphi)$ *iff* $\sigma \notin Sol_R^\Sigma(\varphi)$;
7. $\sigma \in Sol_R^\Sigma(\exists x.\varphi)$ *iff there exists some* $t \in T(\Sigma)$ *such that* $\{x \mapsto t\}\sigma \in Sol_R^\Sigma(\varphi)$; *and*
8. $\sigma \in Sol_R^\Sigma(\forall x.\varphi)$ *iff for every* $t \in T(\Sigma)$, $\{x \mapsto t\}\sigma \in Sol_R^\Sigma(\varphi)$.

Note that this is *not* a set of solutions wrt a theory R, as in [11]; the rewrite system R is only used for the irreducibility constraints (roughly R represents the rewrite rules existing at a finite stage of the completion process). The use of the system $R_{s\sigma \approx s\sigma}$ in case 3, i.e., the set of equations in R with one side smaller than $s\sigma$ and the other side less than or equal to $s\sigma$, is a technical necessity for the induction in the completeness proof, and will be explained below. An alternate way of stating this case, given that $\succ$ is total on ground terms, is that $s\sigma$ is the smallest term in its equivalence class modulo $R_{s\sigma \approx s\sigma}$.

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. Two constraints φ and ψ are said to be *equivalent* if $Sol_R^{\Sigma^+}(\varphi) = Sol_R^{\Sigma^+}(\psi)$ for any R and Σ^+. A constraint is *satisfiable* if there exists some solution for some R and Σ^+, and *unsatisfiable* (and hence equivalent to $\bot$) otherwise; it is *valid* (and hence equivalent to $\top$) if for any R and Σ^+, any ground substitution over Σ^+ is a solution. Clearly, when R is empty, $Irr(s)$ is equivalent to $\top$ for any s. We say that φ is *stronger than* or a *strengthening of* ψ if for every R and Σ^+, $Sol_R^{\Sigma^+}(\varphi) \subseteq Sol_R^{\Sigma^+}(\psi)$; alternately, ψ is *weaker than* or *a weakening of* φ. (Note that "stronger than" and "weaker than" both include the case "equivalent to.") We should emphasize here that the set of solutions to a constraint is not decidable when R is an infinite set (say the ground instances of a finite set of rewrite rules existing at some stage of the completion process). As discussed above, our approach to irreducibility constraints is based on using whatever partial information can be gleaned from the current set of rules to eliminate redundancy and improve the efficiency of the whole process; hence it will not be necessary to have a complete constraint solver. We will discuss this further in section 5.

2.2 Constrained Equations

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. By

[3] Hence, $s\sigma$ will be in normal form with respect to any rewrite system R' contained in $\succ$ and equivalent to $R_{s\sigma \approx s\sigma}$.

$(s \approx t[\![\varphi]\!])\sigma$ we mean $s\sigma \approx t\sigma[\![\varphi\sigma]\!]$, where 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 instance, if we write $f(x) \approx x\,[\![\,x \neq g(y)\,]\!]$, it should be read as $f(x) \approx x\,[\![\,\neg\exists y(x = g(y))\,]\!]$.

Remark In order to preserve completeness, it will be necessary to maintain constrained equations in a certain restricted form. We only allow a constraint of the form $s \approx t[\![\ldots Irr(u)\ldots]\!]$ if either $u \prec s$ or $u \prec t$. For example, this will hold if u is a proper subterm of either s or 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*, then a more refined form of weakening is possible: if 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. The idea, naturally, is to preserve as much of the constraint as possible. We shall assume in the sequel that all equations have this restricted form, in particular, for the sake of clarify, *we shall not mention this weakening process explicitly when forming new equations* in the conclusions of inferences. In addition, it is possible to simplify constraints in other ways (see [11]).

Next we define what is a ground instance of a constrained equation.

Definition 3 *For any ground R contained in $\succ$ and signature Σ, the set of ground instances of an equation $A[\![\varphi]\!]$ relative to R and Σ is defined as*

$$Gr_R^\Sigma(A[\![\varphi]\!]) \;=\; \{\,A\sigma\,|\,Var(A\sigma)=\emptyset \text{ and } \sigma \in Sol_R^\Sigma(\varphi)\}.$$

The set of ground instances relative to R of a set E is then defined as $Gr_R^\Sigma(E) \;=\; \bigcup_{A\in E} Gr_R^\Sigma(A)$.

Hereafter, we omit Σ when it is not relevant.

The *erasure* of an equation $A[\![\varphi]\!]$ is defined as $A[\![\top]\!]$ and similarly for sets of equations. Therefore the set of *all* ground instances of an equation (ignoring the constraint) is $Gr_R^\Sigma(erasure(A))$, but since R serves no purpose, we denote this as $Gr^\Sigma(erasure(A))$. Similarly $Gr^\Sigma(erasure(E)) = \bigcup_{A\in E} Gr^\Sigma(erasure(A))$.

The constraints on equations are used to restrict the ground instances to only those that are useful to complete the set of equations. The other instances of the equations are true in the equational theory, but not necessary for the completion process. Thus, the erasure of a constrained equation represents all the instances of the equation that are true. When the completion process is finished, it may be more convenient to erase the constraints on each equation. We show in our completeness proof that, after erasing the constraints, we still have a canonical rewrite system. In section 8, we discuss how to use our framework in the context of initial constraints. In that case, not all instances of an equation will be true. Therefore, in that section, the concept of erasure is redefined, so it still represents all the instances of an equation that are known to be true. In this way, we give a framework which formally shows the distinction between constraints placed on the equations for efficiency of the completion process and constraints which are necessary to preserve the soundness of the completion procedure.

By the above definition, for some R, a particular equation $A[\![\varphi]\!]$ may have no instances, for example if φ is $Irr(a)$ and $R = \{a \to b\}$. In particular, ground equations with constraints may not have any ground instances! If no irreducibility constraints are present, then naturally R plays no role in defining the ground instances. In general, the constraint will delimit the possible ground instances. For example, if A is $gfy \approx a[\![Irr(fy)]\!]$, then $gfa \approx a$ is an instance relative to $R = \{fb \approx c\}$, but not $gfb \approx a$.

For an inference π in the form
$$\frac{A[\![\varphi_1]\!] \quad B[\![\varphi_2]\!]}{C[\![\varphi_3]\!]}$$
on equations from E, and given a ground set R of unconstrained equations, if $A\tau$, $B\tau$, and $C\tau$ are ground instances relative to R of the premises and conclusion, then we may call
$$\frac{A\tau \quad B\tau}{C\tau}$$
a *ground instance* of π relative to R. By an *inference from ground instances of* E relative to R we mean a ground instance of some inference from E relative to R.

We shall not need the notion of constrained rewriting in this paper, except in the trivial case of sets of ground rewriting systems, and the reader is referred to [11] for details.

3 Redundancy and Constraints

In this paper we present several inference systems for constrained completion. These systems are designed to show the various tradeoffs which can be employed when applying redundancy notions to eliminate certain instances of constrained equations involved in the inferences. The tradeoffs basically arise from considering how to strengthen the constraint attached to one of the equations involved in the inference, possibly at the expense of another constraint. This is a refinement of the kind of constraint weakening which is necessary in the deletion rules of Basic Paramodulation [4, 5, 15]. We also show how previously presented inference systems such as Basic Completion [5, 15] and general superposition [23, 24] can be seen as special cases of this inference system by setting the parameters correctly.

3.1 Redundancy and Correct Equations

Intuitively, a redundant equation is an equation which is implied by smaller equations. Such equations need not play a role in completion, and hence can be removed. This is a semantic version of the well-known *subconnectedness criterion* [3] which encompasses simplification, subsumption and deletion of identities. It was first presented by [1, 17], and our current formulation owes much to the paper [5]. The main differences have to do with the presence of constraints and the fact that we avoid the use of Skolem constants in this paper by talking about redundancy in arbitrary extensions of the given signature.

Definition 4 *Let R be a set of ground rewriting rules and A and B ground equations over a signature Σ. A is R-*redundant below *B* in *E if for every extension Σ^+ there exist ground instances $A_1, \ldots, A_n$ from $Gr_R^{\Sigma^+}(E)$ such that (1) each $A_i \prec B$ and (2) if $R \models A_i$ for each i, then $R \models A$. If A and B are non-ground equations, then A is R-*redundant in *E* below *B if, for every Σ^+, for every grounding substitution σ for A and B such that $A\sigma \in Gr_R^{\Sigma^+}(A)$ and $B\sigma \in Gr_R^{\Sigma^+}(B)$, $A\sigma$ is R-redundant in E below $B\sigma$. If B is omitted it is assumed to be A and E will be omitted if it is available from context. If A is R-redundant for any canonical R, then A 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* up to *M if, for every Σ^+, for each grounding substitution σ for A and M such that $A\sigma \in Gr_R^{\Sigma^+}(A)$ and $B_i\sigma \in Gr_R^{\Sigma^+}(B_i)$ for each i, there exist ground instances $A_1, \ldots, A_n$ from $Gr_R^{\Sigma^+}(E)$ such that (1) each $A_i \preceq max(B_1\sigma, \ldots, B_k\sigma)$ and (2) if $R \models A_i$ for each i, then $R \models A$.*[4]

For instance, equations all of whose ground instances are identities and equations with unsatisfiable constraints are trivially redundant. Some more interesting examples of redundant

[4]The point of this rather complex notion will be made clear following the next definition.

equations may perhaps clarify the definition. The equations $f(x) \approx x[\![x = a \wedge x \neq a]\!]$ and $f(a) \approx b[\![Irr(a) \wedge \neg Irr(a)]\!]$ are both redundant because they have no ground instances. The equation $f(x) \approx f(x)$ is redundant because it is an identity and therefore implied by the empty set of equations. The equation $f(a) \approx b$ is redundant if the equations $a \approx c$ and $f(c) \approx b$ exist in E (where $a \succ c$), because they imply $f(a) \approx b$. The equation $f(a) \approx b[\![Irr(b)]\!]$ is redundant if the equations $a \approx c$ and $f(c) \approx b[\![Irr(b)]\!]$ exist in E, because any R which makes true all instances of the smaller equations also makes true $f(a) \approx b[\![Irr(b)]\!]$. The equation $f(a) \approx b[\![Irr(a)]\!]$ is redundant if the equation $a \approx c$ is in E because any equation which makes $a \approx c$ true will reduce a and therefore $f(a) \approx b[\![Irr(a)]\!]$ has no instances.

A comment on the use of the extended signature Σ^+ is in order. The usual method of proving that the result of an ordered completion process is canonical (and not just that the set of orientable instances is ground canonical) is to add a set of Skolem constants to the signature during the completeness proof. Then it is shown that no property of the constants was used during the inference process or the completeness proof. In our setting, it is important to be precise about the signature vis a vis constraints, since we do not want to delete an equation whose constraint is unsatisfiable in the given signature, but satisfiable with the addition of Skolem constants. For this reason, we consider a more complex definition of redundancy which accounts for extensions to the signature. This explains the transition from ground canonical to canonical in a more fundamental way.

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. In this framework a constrained equation will be represented by 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., $Gr_R^{\Sigma}(A[\![\varphi_1, \varphi_2, M]\!]) = Gr_R^{\Sigma}(A[\![\varphi_1]\!])$. The other constraint and the set M record redundancy information in the following way.

Definition 5 *Let E be a a set of constrained equations over Σ. Then $A[\![\varphi_1, \varphi_2, M]\!] \in E$ is* correct *in E (or simply* correct *if E is obvious) if for all R and every extension Σ^+*

1. $Gr_R^{\Sigma^+}(A[\![\varphi_1]\!]) \subseteq Gr_R^{\Sigma^+}(A[\![\varphi_2]\!])$,
2. *If $B \in Gr_R^{\Sigma^+}(A[\![\varphi_2]\!]) \setminus Gr_R^{\Sigma^+}(A[\![\varphi_1]\!])$ then B is R-redundant in E, and*
3. *If $B \in Gr^{\Sigma^+}(erasure(A))$ then B is R-redundant in E up to M.*

For example, an unconstrained equation is written in correct 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 they are not relevant.

The last two components are used to store information about the history of an equation. When an equation is created, $\varphi_1 = \varphi_2$. Irreducibility constraints may be added to φ_1 to indicate that variables may be assumed to be in normal form. At that point, and at every point after that, φ_1 is stronger than φ_2. Then, when inferences are performed, the constraints on φ_1 are further strengthened, because certain instances of the equation have been simplified and are no longer needed. Throughout the completion process, the constraint φ_1 becomes stronger while φ_2 remains the same, as more redundant instances are removed from φ_1. Thus, an equation can use the constraint φ_2 when it is being used as a simplifier, because φ_2 allows more instances to be available to simplify another equation. The component M is also used to allow more instances to be used as a simplifier. M is the set of original axioms which are ancestors of the equation. Since we are working with initially unconstrained equations, we know that all instances of an equation are true, not just the instances represented by φ_1 and φ_2. Therefore, M is a set of unconstrained equations which imply all instances of A. If an equation bigger than all equations in M is simplified by A, then we know that all instances of A are available to perform the simplification.

Essentially, redundancy is used in the completeness proof to show at what point in the induction equations become true. If an equation is implied by smaller equations, then it is not needed in constructing the canonical rewrite system. 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 $R \models A_i$..."). Thus it is useful to know when the logical requirements are satisfied by a set of equations smaller than some other equation B. In practice, it is only necessary to consider axioms (original equations) involved in the construction of A during the completion process, and so M records the history of the equation in this way. Any equation is true in a model constructed to satisfy its associated set M. In practice we only need to save the maximal elements in M, so M could be replaced with a smaller set, $max(M)$. This parameter does not change once a particular equation is constructed.

This paper is primarily concerned with techniques for modifying constraints during the process of completion, in order to take advantage of redundancy information. We will denote such a *constraint modification* by the notation

$$A\,[\![\,\varphi_1, \varphi_2\,]\!] \Rightarrow A\,[\![\,\psi_1, \psi_2\,]\!],$$

where sometimes we omit the second occurrence of A for simplicity. The main idea used in this paper is that such transformations can be performed whenever we only delete redundant instances of equations, or add instances which are provable. We may formalize this as follows.

Definition 6 *Let E be a set of constrained equation over Σ. A constraint modification $A\,[\![\,\varphi_1, \varphi_2\,]\!] \Rightarrow A\,[\![\,\psi_1, \psi_2\,]\!]$ is* correct *in E (or simply* correct *if E is obvious) if for all R and Σ^+,*

1. *Every member of $Gr_R^{\Sigma^+}(A\,[\![\,\varphi_1\,]\!]) \setminus Gr_R^{\Sigma^+}(A\,[\![\,\psi_1\,]\!])$ is R-redundant in E;*
2. *$Gr^{\Sigma^+}(erasure(E)) \models Gr_R^{\Sigma^+}(A\,[\![\,\psi_1\,]\!]) \setminus Gr_R^{\Sigma^+}(A\,[\![\,\varphi_1\,]\!])$; and*
3. *Every member of $Gr_R^{\Sigma^+}(A\,[\![\,\psi_2\,]\!]) \setminus Gr_R^{\Sigma^+}(A\,[\![\,\varphi_2\,]\!])$ is R-redundant in E.*

The point of the first condition is that we only delete redundant equations, the second says that any instances added must be consequences of the underlying equational theory, and the last requires that any weakening of the second constraint only involve the adding of additional redundant instances. This insures that if an equation is correct, then such a modification will produce another correct equation.

Lemma 1 *Let E be a set of equations over Σ containing $A\,[\![\,\varphi_1, \varphi_2\,]\!]$. Consider the constraint modification $A\,[\![\,\varphi_1, \varphi_2\,]\!] \Rightarrow A\,[\![\,\psi_1, \psi_2\,]\!]$, and let E' be $(E \setminus A\,[\![\,\varphi_1, \varphi_2\,]\!]) \cup A\,[\![\,\psi_1, \psi_2\,]\!]$. If the modification is correct in E and $A\,[\![\,\varphi_1, \varphi_2\,]\!]$ is correct in E then $A\,[\![\,\psi_1, \psi_2\,]\!]$ is correct in E and, for all R and Σ^+, $Gr_R^{\Sigma^+}(E')$ is formed by deleting redundant equations from $Gr_R^{\Sigma^+}(E)$ and adding equations that follow from $Gr^{\Sigma^+}(erasure(E))$.*

Proof. This follows immediately from the definition of *correct* constraint modification. □

The framework introduced in this subsection for expressing redundancy notions, and for transforming constraints, will be used to delete instances of equations during the completion process. We can also make explicit redundancy notions which are usually left implicit in the form of the inference system. For example, it is well known that in paramodulation and completion, overlaps at variable positions are not necessary for completeness. The technical reason for this is that a ground instance of a clause or equation which is reducible at a substitution position is redundant, since we can normalize the substitution terms, and this smaller instance, with the equations used to do the normalization, imply the original instance. In our framework we can make this explicit, representing the irreducibility condition in the constraint. An unconstrained equation $fx \approx gx$ would be represented in correct form here as $fx \approx gx \llbracket Irr(x), \top \rrbracket$, which says that any binding for x must be irreducible. Note that we can not in general in completion say when such a constraint is true, because we are evolving successive approximations of a limit canonical system, but it is sufficient to consider cases where the constraint is false, to simulate the "no overlaps at variable positions" condition and also significant extensions to the Basic strategy.

3.2 Equation Sets with Initial Constraints

In the main body of this paper, we consider only the problem of completing an initially unconstrained set of equations, where constraints are added during completion to record information about redundant instances of equations. The goal is to make the process more efficient. However, sometimes initial constraints are useful to express information in a compact form (cf. [11]), and so it is worth extending our framework to this situation and understanding what is required. In addition, it turns out that the standard "no variable overlaps" condition can be naturally expressed using the irreducibility predicate. Thus we will present the general framework for initial constraints now, although in the rest of the paper (except for section 8) we assume only a very simple form of initial constraint to represent the "no variable overlaps" condition. In section 8 we will return to consider the more general case of initial constraints.

Initial constraints are problematic, as it is well–known that some sets of equations with initial constraints are not completeable without overlapping at variable positions or performing some inferences involving the constraints themselves. For example, $fx \approx gx \llbracket x \neq b \rrbracket$ and $a \approx b$, where $f \succ g \succ a \succ b$, is not canonical (since fb and gb have no rewrite proof), but no overlap exists at a non-variable position. The fundamental problem is that the restriction of a paramodulation-type inference system to non-variable positions depends on showing that ground instances which are reducible inside a substitution term are redundant. The idea is that if we normalize the substitution terms, then the reduced instance and the reducing equations are smaller than, and imply, the original instance. When we are using constraints, we must know that these reduced instances are available. This is not so in our example above, and so we can not show that all ground instances of the first equation which are reducible at a substitution position are redundant, which in turn means we can not restrict inferences to non-variable positions and preserve completeness. In our framework, this means we can not represent this equation with initial constraint in the form $fx \approx gx \llbracket Irr(x) \wedge x \neq b, x \neq b \rrbracket$, since $fa \approx ga$ is not redundant, as the reduced instance $fb \approx gb$ is not available. We would need to make a weaker assertion than $Irr(x)$, namely that x is irreducible if b is not, namely, $fx \approx gx \llbracket (Irr(x) \vee Irr(b)) \wedge x \neq b, x \neq b \rrbracket$. When the system contains just the two equations given, then b is irreducible and we can not assume x is bound to an irreducible term, and hence must allow a variable overlap. Suppose further that the unconstrained equation $b \approx c$ exists, so that $Irr(b)$ is not true. Then we need not perform a variable overlap, since we know that $Irr(x)$ is true. As a practical matter, since we can only say when an irreducibility constraint is false in an evolving system, this means that when initial constraints are present, we only have a criterion for eliminating variable overlaps. In any case, we can state very precisely the relationship between initial constraints and the necessity for variable overlaps, at the cost of some technical machinery which we now introduce.

We will define a binary function $IrrCon$ from Irr which will take a variable and a constraint as arguments and produce a new constraint which shows which irreducibility conditions can be added to the constraint so that some redundant instances of the equation have been deleted because they are reducible. Then we will define a function called $IrrVar$ which will recursively allow us to add these irreducibility constraints. We will also define a function called $NoIrrVar$ which will allow us to add new instances of an equation which are redundant by the given instances.

Definition 7 *Let $\mathcal{V}$ be the set of all variables. Then define $IrrCon : \mathcal{V} \times \mathcal{C} \longrightarrow \mathcal{C}$ such that*

$$IrrCon(x, \varphi) = Irr(x) \vee \exists y(Irr(y) \wedge \neg\varphi[x \mapsto y]).$$

The function $IrrCon$ takes a variable x and a constraint φ as arguments. It returns a new constraint saying that x is irreducible if every irreducible substitution for x satisfies φ. The idea behind this is that we can delete reducible instances of the equation if they are redundant by virtue of available irreducible instances. This generalizes the "no overlaps at variable positions" of ordinary paramodulation and completion to the case of initial constraints. Note that all reducible instances will be redundant if all irreducible instances satisfy φ.

For example, $IrrCon(x, \top) = Irr(x) \vee \exists y(Irr(y) \wedge \neg\top) = Irr(x) \vee \exists y(Irr(y) \wedge \bot) = Irr(x) \vee \bot = Irr(x)$. Since there are no irreducible instances of x which do not satisfy x, we can remove the reducible instances of x in the constraint $\top$.

Following the example at the beginning of this subsection, $IrrCon(x, x \neq b) = Irr(x) \vee \exists y(Irr(y) \wedge y = b)$ which is equivalent to $Irr(x) \vee Irr(b)$. If x is not irreducible then there must be irreducible substitution for x which does not satisfy the constraint. But since b is the only value of x which does not satisfy the constraint, we can say that b is irreducible if x is not irreducible.

Now we define the functions $IrrVar$ and $NoIrrVar$ which tell us how to modify the constraints in an equation to delete these redundant instances. The function $IrrVar$ takes a set of variables S and a constraint φ as an argument. The output of the function is a new constraint ψ which represents φ conjoined with the condition that each variable in S is irreducible as long as there is no substitution for that variable which is irreducible and does not satisfy φ. This guarantees that ψ removes only redundant instances from the equation constrained by φ.

The function $NoIrrVar$ does exactly the opposite of $IrrVar$. It takes a set of variables S and a constraint φ as inputs. Its output is a constraint ψ which represents the instances of the equation constrained by φ which are represented by φ or redundant by the instances represented by φ. The constraint ψ adds values for each variable x in S which are reducible, as long as we can be guaranteed that the normal form of x satisfies the constraint. This will be true as long as there are no irreducible instances which do not satisfy the constraint.

The functions $IrrVar$ and $NoIrrVar$ are just recursive applications of the function $IrrCon$. By abuse of notation, we also extend $IrrVar$ and $NoIrrVar$ to allow an equation as the first argument. In that case, it is interpreted as meaning the variables in that equation.

Definition 8 *Let $\mathcal{E}$ be the set of all equations. Define $IrrVar : 2^V \times C \longrightarrow C$ recursively so that*

- *$IrrVar(\emptyset, \varphi) = \varphi$ and*
- *$IrrVar(\{x\} \cup S, \varphi) = IrrVar(S, \varphi) \wedge IrrCon(x, IrrVar(S, \varphi))$.*

Similarly define $NoIrrVar : 2^V \times C \longrightarrow C$ so that

- *$NoIrrVar(\emptyset, \varphi) = \varphi$ and*
- *$NoIrrVar(\{x\} \cup S, \varphi) = NoIrrVar(S, \varphi) \vee \neg IrrCon(x, NoIrrVar(S, \varphi))$.*

We also define $IrrVar, NoIrrVar : E \times C \longrightarrow C$ so that

- *$IrrVar(A, \varphi) = IrrVar(Var(A), \varphi)$ and*
- *$NoIrrVar(A, \varphi) = NoIrrVar(Var(A), \varphi)$.*

Given the examples above, it is easy to see that $IrrVar(\{x\}, \top) = Irr(x)$ and $IrrVar(\{x\}, x \neq b) = (x \neq b) \wedge (Irr(x) \vee Irr(b))$. In both cases, the only instances removed were redundant. Also, the reverse holds true, i.e., $NoIrrVar(\{x\}, Irr(x)) = \top$ and

$$NoIrrVar(\{x\}, x \neq b \wedge (Irr(x) \vee Irr(b))) \quad = \quad (x \neq b).$$

In both cases, only redundant instances were added. This formalizes the constraint modification which we discussed at the beginning of this subsection.

Now we can show that the constraint modification which removes all reducible instances of an equation is correct.

Lemma 2 $A[\![\, \varphi, \varphi \,]\!] \Rightarrow A[\![\, IrrVar(A,\varphi), NoIrrVar(A,\varphi) \,]\!]$ *is a correct constraint modification.*

Proof. First note that for every variable x and constraint $\varphi[x]$, the meaning of $IrrCon(x,\varphi)$ is that the binding for x is irreducible or there is some irreducible term t which does not satisfy the constraint $\varphi[x]$. Therefore any $A\sigma \in A[\![\, \varphi \wedge IrrCon(x,\varphi) \,]\!]$ is a ground instance of A such that σ satisfies φ and either $x\sigma$ is irreducible or there is an irreducible instance $A\theta$ such that $\varphi\theta$ is unsatisfiable. For every R, if C is an instance in $Gr_R(A[\![\, \varphi \,]\!]) \setminus Gr_R(A[\![\, \varphi \wedge IrrCon(x,\varphi) \,]\!])$ then C is reducible by R, and in addition all irreducible instances of A satisfy φ. Therefore C reduces to an instance that satisfies φ. This means that C is redundant in any E containing $A[\![\, \varphi \wedge IrrCon(x,\varphi) \,]\!]$.

Therefore all members of $A[\![\, \varphi \,]\!]$ are redundant in any E containing $A[\![\, IrrVar(A,\varphi) \,]\!]$, and all members of $A[\![\, NoIrrVar(A,\varphi) \,]\!]$ are redundant in any E containing $A[\![\, \varphi \,]\!]$. Therefore all members of $A[\![\, NoIrrVar(A,\varphi) \,]\!]$ are redundant in any E containing $A[\![\, IrrVar(A,\varphi) \,]\!]$. $\Box$

This shows us that for any equation A and constraint φ we can replace the equation $A[\![\, \varphi \,]\!]$ by the equation $A[\![\, IrrVar(A,\varphi), NoIrrVar(A,\varphi) \,]\!]$, because to do so gives us a new correct equation and it only removes redundant instances of the original equation.

For example the following constraint modifications are correct.

- $f(x,y) \approx f(y,x)[\![\, \top, \top \,]\!] \Rightarrow [\![\, Irr(x) \wedge Irr(y), \top \,]\!]$
- $f(x,y) \approx f(y,x)[\![\, Irr(x) \wedge Irr(y), \top \,]\!] \Rightarrow [\![\, Irr(x) \wedge Irr(y), \top \,]\!]$
- $f(x) \approx g(x)[\![\, Irr(x) \wedge x \neq b, \top \,]\!] \Rightarrow [\![\, Irr(x) \wedge x \neq b, (Irr(x) \wedge x \neq b) \vee Irr(b) \,]\!]$

Such explicit irreducibility constraints make the role of initial constraints clearer. In section 8, we present a method for introducing explicit irreducibility constraints into equations with initial constraints and show how the inference system would need to be relaxed in order to allow a limited form of variable overlap. Until then, however, we shall confine ourselves to initially unconstrained sets of equations. Such equations are represented in correct form as $A[\![\, Irr(x_1) \wedge \ldots \wedge Irr(x_n), \top \,]\!]$, where $\{x_1, \ldots, x_n\} = Var(A)$. We assume all initial sets of equations have this form.

4 Constrained Critical Pair Generation

In this section we give a generalization of the critical pair rule from [11] for ordered completion and show how a variety of tradeoffs may be obtained in deleting various instances of the equations involved in an inference of this form. We will prove that the inference rules are sound and that all equations created are correct. Then we discuss the manner in which the constraints of the equations involved may be modified. The basic idea is to delete instances of the right premise which are redundant by virtue of instances of the conclusion and the left premise, and there is a tradeoff between how many instances of the right premise to delete and how many instances of the other equations are made available. We will present several possibilities and prove their correctness. In section 6, we will prove that the completion procedure is complete, i.e., that a ground canonical set of equations is generated in the limit.

4.1 The C-Deduce Inference Schema

The following inference schema characterizes the class of critical pair rules we consider in this paper.

C-Deduce

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

where

1. $s\sigma \not\preceq t\sigma$,
2. $u[s']\sigma \not\preceq v\sigma$,
3. $u[s']\sigma \approx v\sigma \not\preceq s\sigma \approx t\sigma$
4. $\sigma = mgu(s, s')$,
5. $\Delta_1 \;=\; \varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(T)$, where $T = \{w \mid w \text{ is a proper subterm of } s\sigma\} \cup \{w \mid w \text{ is a subterm of } u\sigma \text{ and } w <_r s'\sigma\}$,
6. $\Delta_2 \;=\; NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$.

In addition, this schema specifies that we may perform any *correct constraint modifications* on the conclusion and the premises, subject to the following restrictions:

- We may weaken Δ_1, but *only* to a constraint that is still a strengthening of $IrrVar(u[t]\sigma \approx v\sigma, \top)$ (so that we still forbid inferences into variable positions); and
- We may strengthen Δ_2, but *only* to a constraint that is a weakening of Δ_1 (else the conclusion is no longer correct).

Specific instances of this general schema may involve specific constraint modifications; examples will be given below.

The constraint Δ_1 represents the strongest possible conditions we can have to make the completeness proof go through. In general in the inference rules we present, equations will have the form $A \,[\![\, Irr(s_1) \wedge \cdots \wedge Irr(s_n) \wedge \varphi_1', \varphi_2, M \,]\!]$, where any variable in A occurs in some s_i, as mentioned above. That condition already holds if Δ_1 is not weakened, because all premise equations have the constraint that their variables are irreducible, and all variables in the conclusion are from the premises. 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 constraint Δ_2 allows us to add some redundant instances for the second constraint on the premise. For instance, if $\Delta_1 = Irr(x) \wedge Irr(y)$ then $\Delta_2 = NoIrrVar(\Delta_1) = \top$. As long as Δ_2 remains a weakening of Δ_1, we are guaranteed that the conclusion is correct. Performing more constraint modifications will allow us to apply some simplifications to the equations in the inference, as long as only redundant instances are deleted.

If we had ordering constraints, we could add the first three conditions of C-Deduce as constraints in the the conclusion of the inference, with the $\not\preceq$ replaced by $\succ$. This would allow us to hereditarily guarantee that the ordering conditions hold (see [16]).

The following lemma assures us that any instances of the C-Deduce schema is sound and that it produces a correct equation.

Lemma 3 *Let E be a set of equations over Σ. If the premises of an inference which is an instance of C-Deduce are correct equations in E then, for every R and Σ^+, every R-instance of the conclusion is implied by some equations in $Gr_R^{\Sigma^+}(E)$. Also, the conclusion is a correct equation in E.*

Proof. The reader will verify that the inference is sound. To prove the correctness of the conclusion, we note that, for all R and Σ^+, all instances in $Gr_R^{\Sigma^+}(u[t]\sigma \approx v\sigma [\![\Delta_2]\!]) \backslash Gr_R^{\Sigma^+}(u[t]\sigma \approx v\sigma [\![\Delta_1]\!])$ are redundant. This follows from lemma 2 and the fact that Δ_2 is a strengthening of $NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$. Also, all σ-instances of both premises are true by equations smaller than some equation in $M\sigma \cup N\sigma$. Therefore all instances of the conclusion are true by equations smaller than some equation in $M\sigma \cup N\sigma$. □

In section 4.2, we will show how it is possible to remove redundant instances of the right premise. In addition, it is possible to define situations under which the inference itself is redundant and hence need not be performed. Before we define this notion, we need to define a ground version of our inference rule. The *Ground-Deduce* rule is simply a superposition of the form

$$\frac{s \approx t \qquad u[s] \approx v}{u[t] \approx v}$$

where $s \succ t$, $u[s] \succ v$, and $u[s] \approx v \succ s \approx t$. Such an inference is called a ground instance modulo R of a C-Deduce inference if there exists a σ which is an R-solution to, and grounds, the premises and the conclusion, and maps the inference onto the Ground-Deduce inference.

Definition 9 *For any E and ground R, a Ground-Deduce inference as given above is R-*redundant *if (i) either premise is R-redundant, or (ii) the conclusion is R-redundant below $u[s] \approx v$. A C-Deduce inference is R-redundant if every ground instance wrt R is R-redundant. If an inference is R-redundant in E for every R, then it is simply called redundant in E.*

4.2 Constraint Modification

The inference rules we present are instances of the C-Deduce schema presented above; to define an inference rule, then, we need to say what the constraints in the conclusion are, and how the constraints in the premises are (potentially) modified. For each case, we must show that the conditions of C-Deduce are satisfied.

First we present two general constraint modification rules, *CM1* and *CM2*, that may be applied to strengthen the right premise after an inference has been performed. The general idea of these inference rules is that after the conclusion has been added to the set of equations, some instances of the right premise are now redundant. The constraint modification rules allow us to remove those redundant instances. 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 critical pair rules, or as refinements of the special simplification rule used in Basic Paramodulation [4, 5, 15].

To see which instances are redundant, we give the following lemma, which is proved by a simple application of the definition of correct constraint modification.

Lemma 4 *Let $A [\![\varphi]\!]$, $A_1 [\![\varphi_1]\!], \cdots, A_n [\![\varphi_n]\!]$ be correct equations in E. Let σ be a substitution such that, for all grounding substitutions τ, $A_1\sigma\tau, \cdots, A_n\sigma\tau \models A\sigma\tau$, and for all i, $A_i\sigma\tau \prec A\sigma\tau$. Then all instances of $A [\![\varphi \wedge \sigma \wedge \varphi_1 \wedge \cdots \wedge \varphi_n]\!]$ are redundant. Therefore $A [\![\varphi]\!] \Rightarrow [\![\varphi \wedge \neg(\sigma \wedge \varphi_1 \wedge \cdots \wedge \varphi_n)]\!]$ is a correct constraint modification.*

Definition 10 *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: $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge \Delta_2)$.*

For example, suppose that σ is the substitution $[x \mapsto a, y \mapsto fa]$, ψ_1 is the constraint $\top$, Δ_2 is the constraint $\top$, and φ_2 is the constraint $Irr(gx)$. Then CM1 modifies ψ_1 to $x \neq a \vee y \neq fa \vee \neg Irr(gx)$, which can be simplified to the equivalent constraint $x \neq a \vee y \neq fa \vee \neg Irr(ga)$. If y does not appear in the equation constrained by ψ_1, then ψ_1 can be simplified to $x \neq a \vee \neg Irr(ga)$. The constraint modification CM2 modifies ψ_1 to $x \neq a \vee y \neq fa$. If y does not appear in the equation constrained by ψ_1, then ψ_1 can be simplified to $x \neq a$.

If $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$ and $t\sigma \prec s\sigma$ then CM1 applied to the right premise simply deletes instances of the right premise that are redundant due to instances of the left premise that exist, instances of the left premise that are redundant, and instances of the conclusion. Therefore the resulting right premise is a correct equation.

Lemma 5 *If $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$ and $t\sigma \prec s\sigma$ then CM1 is a correct constraint modification.*

Proof. We only need to apply the previous lemma, since Δ_2 represents instances of the conclusion that are true by equations equal to or smaller than it, and φ_2 represents instances of the left premise that are true by equations equal to or smaller than it. $\square$

The only way the condition that $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$ can be violated is if the overlap is at the root and $t\sigma$ and $v\sigma$ are not comparable, in which case the conclusion will not be orientable; in standard completion this causes an immediate failure. The condition that $t\sigma \prec s\sigma$, also, cannot be violated in standard completion. Therefore, in standard completion, the constraint modification CM1 is always possible. In ordered completion, it is necessary to check these conditions.

If, in addition to the above conditions, $M\sigma \prec_{mul} \{u[s']\sigma \approx v\sigma\}$, then CM2 applied to the right premise deletes instances of the right premise that are redundant due to instances of the conclusion and some instance of the left premise. This can be any instance of the left premise, not just the instances that exist. This is because the third constraint on the left premise guarantees that all instances of the equation are true by equations smaller than the right premise. Therefore the resulting right premise is correct after the constraint modification.

Lemma 6 *If $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$, $t\sigma \prec s\sigma$, and $M\sigma \prec_{mul} \{u[s']\sigma \approx v\sigma\}$, then CM2 is a correct constraint modification.*

Proof. The proof here is the same as the proof of the previous lemma, except we need the fact that all σ-instances of the left premise are true by equations smaller than or equal to some equation in $M\sigma$. $\square$

4.3 CCP Rule

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

Definition 11 *Let CCP be the instance of C-Deduce where $\Delta_1 = \varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(T)$, $\Delta_2 = NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$, and where CM1 is performed if $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$ and $t\sigma \prec s\sigma$.*

Now we consider some examples of this rule. In all the examples for the rest of section 4, we will assume the following convention for each constraint $A[\![\varphi_1, \varphi_2, M]\!]$ with missing parameters. 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 $\top$. Consider the inference

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

on axioms. If we use the CCP rule then we may apply CM1 to the constraint of the right premise as follows: $Irr(x) \Rightarrow Irr(x) \wedge (x \neq a \vee \neg Irr(a))$.

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

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

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 $fa \approx ga$ is used as a left premise, fa can be restricted to be in normal form (cf. the prime superposition criterion discussed in section 4.5), which violates the constraint.

The point of our approach is to try to restrict the available instances while at the same time storing information about which instances are redundant. In the CCP inference Δ_1 is as strong as it can be in an inference. Given the value of Δ_1 we have made Δ_2 as weak as possible so we can delete more of the instances of the right premise. For example, if $\Delta_1 = \top$ then $\Delta_2 = Irr(x)$. In general, if $\Delta_1 = Irr(x) \wedge \varphi'$ and φ' does not further constrain x, then $\Delta_2 = \varphi'$ (this process is iterated). The application of CM1 at the end of the inference step will delete as many instances of the right premise as possible, given that the emphasis here is on strengthening the conclusion.

4.4 C-Simplify Rule

Our second rule based on C-Deduce emphasizes strengthening the constraint of 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 12 *C-Simplify is the instance of C-Deduce such that $\Delta_1 = \psi_1\sigma$, $\Delta_2 = NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$, and where in addition if $s\sigma \approx t\sigma \prec u[s']\sigma \approx v\sigma$ and $t\sigma \prec s\sigma$, then we change ψ_1 in the right premise to $\psi_1 \wedge \neg\sigma$; finally, unless $M\sigma \prec_{mul} \{u[s']\sigma \approx v\sigma\}$ 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 $\varphi_2 \Rightarrow \varphi_2 \vee (\sigma \wedge \psi_1)$.*

Consider the first example given above, except as a C-Simplify inference instead of a CCP inference:

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

Applying CM1, the constraint of the right premise is modified as follows: $Irr(x) \Rightarrow Irr(x) \wedge x \neq a$.

Comparing this with CCP, we see that C-Simplify has given the right premise a stronger constraint, but the conclusion has a weaker constraint.

We can now show how constraints provide additional information usable in later inferences. Assume we followed the C-Simplify inference just given with

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

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

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

If we want this to be a C-Simplify inference the conclusion can be weakened to

$$fb \approx ga\,[\![\, \top, \top, fga \approx ga \,]\!].$$

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.

In C-Simplify, we perform the CM1 or CM2 constraint modification. But, before CM1 is performed it may be necessary to add some instances of the left premise. We are always allowed to add instances to an equation, since we assume our initial set of equations is unconstrained, with the result that all instances of each equation are true. Therefore the above constraint modifications are instances of CM1 and CM2.

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

The reader will note that these constraint modifications are instances of CM1 and CM2.

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.[5] Thus it is possible to define a spectrum of possible critical pair rules in our framework.

In this section we have presented critical pair rules for constrained completion. In order to implement a completion procedure based on these rules, we would need to embed these in a comprehensive set of inference rules such as described in [3]. This is a relatively straightforward adaptation of the ideas above, except that deletion rules are formalized in our framework as blocking rules, which are presented in section 5. Irreducibility constraints give us blocking rules based on the reducibility of terms in constraints $Irr(s)$. For example, suppose we have equations $A\,[\![\cdots Irr(u[s'])\cdots]\!]$ and $s \approx t[\![\,\varphi\,]\!]$, where $s\rho = s'$ and $s\rho \succ t\rho$. Then the first equation can be changed to $A\,[\![\cdots(Irr(u[s'])\wedge\neg(\varphi\rho))\cdots]\!]$. Clearly if all instances of $s \approx t$ are available, i.e., $\varphi = \top$,

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

then this corresponds to solving the constraint $Irr(u)$ by replacing it with $\perp$. In cases where only certain instances of $s \approx t$ are available, then we can only falsify the irreducibility constraint for those instances of s'. Such blocking rules are closely related to constraint solving techniques. See section 5 for a discussion.

4.5 Relationship to Critical Pair Criteria

We have indicated in the introduction that this technique of constrained critical pairs covers all known critical pair criteria. 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 completion without constraints.

The standard critical pair rule can be represented in our system by letting $\Delta_1 = Irr(x_1) \wedge \cdots \wedge Irr(x_n)$, where $\{x_1, \ldots, x_n\} = Var(u[t]\sigma \approx v\sigma)$, and $\Delta_2 = \top$. This is only necessary to disallow superposition into variable positions. The standard 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 $\perp$ so the equation may be deleted.

Prime superposition [10] is a critical pair criterion which states that an inference is unnecessary if the left hand side 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 one result of our inference system is a hereditary version of prime superposition because that constraint is kept with the equation and passed along in future inferences.

General superposition [24] and the critical pair criteria discussed in [12, 21, 22] 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 (unconstrained) 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 instances are no longer present. In fact our inference system suggests a hereditary version of general superposition, allowing constraints to be passed from the premises of an inference to the conclusion. 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. Note that this is only true when CM1 can be performed, i.e., when the ordering conditions hold. In that case, we say that CM1 is *possible*.

In addition to naturally simulating subsumed critical pair criteria with our inference system we also naturally simulate *Basic Completion* [5, 15]. In Basic Completion each equation consists of a skeleton and a substitution. In this strategy, when an inference is performed the mgu is saved with the substitution and not applied to the skeleton. Then we never overlap a position generated by substitution; this is a stronger, hereditary version of the "no variable overlaps" restriction. We simulate Basic Completion with irreducibility constraints. In the conclusion of an inference we let $\Delta_1 = \varphi_1\sigma \wedge \psi_1\sigma$ and $\Delta_2 = NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$. 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. Thus the "frontier" of the terms (to use the terminology of [5]) is presented by the terms in the constraints. A blocking rule is necessary to implement the restriction, naturally. In fact, this is stronger than the formulation from [4, 5] because we are allowing irreducibility constraints on terms that do not appear in the equation; as long as they are smaller than some side of the conclusion, they can be inherited from the premises and preserved. If we let $\Delta_1 = \varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(T)$, where $T = \{w | w \text{ is a proper subterm of } s\sigma\} \cup \{w | w \text{ is a subterm of } u \text{ and } w <_r s'\sigma\}$ we simulate Basic Completion with selection rules and redex orderings, a stronger form of Basic Completion from [5], because in that case we have added the hereditary version of prime superposition.

Basic Completion needs a special form of simplification to be complete, as shown in [4, 5, 15]. In Basic simplification only the substitution positions of the right premise are allowed to be substitution positions of the conclusion. Then certain skeleton positions of the left premise must be instantiated by the substitution before simplification is performed. The only skeleton positions which do not need to be instantiated are the ones which also appear in the substitution of the right premise. For our simulation of Basic simplification we set $\Delta_1 = \psi_1\sigma$ and $\Delta_2 = NoIrrVar(u[t]\sigma \approx v\sigma, \Delta_1)$. Then we weaken the constraints on the left premise by: $\varphi_1 \Rightarrow \varphi_1 \vee (\sigma \wedge \psi_1)$ and $\varphi_2 \Rightarrow \varphi_2 \vee (\sigma \wedge \psi_1)$. Finally we perform CM1. Since σ matches s to s' the result of CM1 will be $\psi_1 \Rightarrow \perp$ and the right premise may be deleted. A stronger version of Basic Completion would be to use CM2 which would give us more cases where the terms in the left premise need not be instantiated in order to weaken ψ_1 to $\perp$. Also, we need not require that σ be a matching substitution. In that case we would simplify some instances of the right premise. We can also formulate the blocking rule from [5] in our framework. We will expand upon this in the next section.

In the conclusion we will discuss the further relationship of our work with other papers such as [11].

5 Constraint Solving

Since the constraints are used to determine when equations and inferences are redundant, we need a constraint solving algorithm. This algorithm does not need to be complete in the sense that it always determines whether or not a constraint is satisfiable, because the only result of an incomplete algorithm is that some redundant equations won't be deleted and some redundant inferences will be performed. The more unsatisfiable constraints a constraint solving algorithm could detect, the more useful it will be. However, the algorithm does need to be sound in the sense that it should not say that a constraint is unsatisfiable when it is satisfiable, because that would cause the system to delete equations that are not redundant and not perform necessary inferences.

Whether we view this as an actual completion procedure or a formalism for describing and proving complete other completion procedures, it is necessary to develop some techniques for constraint solving. First we show, for every term t and set of equations E, a constraint ψ containing only equational and disequational predicates, such that t is reducible in E if ψ is true. So then, given a set of equations E, we transform a predicate $Irr(t)$, appearing in a constraint φ, into the conjunction $Irr(t) \wedge \neg\psi$. Given a constraint solver working on only equality and disequality constraints, we describe below how to give a new constraint solver that tries to solve φ but ignores the irreducibility predicate. This will give us a generalization of the blocking rule from [5].

Lemma 7 *Let E be a set of equations containing the rewrite rules $\{s_1 \to t_1 [\![\varphi_1]\!], \cdots, s_n \to t_n [\![\varphi_n]\!]\}$. Let $A [\![\cdots Irr(u) \cdots]\!] \in E$. Let $\{u_1, \cdots, u_m\}$ be the set of all subterms of u. Let $\psi = \bigvee_{1 \le j \le m} \bigvee_{1 \le i \le n} (\varphi_i \wedge (u_j = s_i))$. Then $A [\![\cdots Irr(u) \cdots]\!] \Rightarrow A [\![\cdots (Irr(u) \wedge \neg\psi) \cdots]\!]$ is a correct constraint modification.*

Proof. To show the correctness of the constraint modification, we will show that for any R and Σ^+, every member of $M = Gr_R^{\Sigma+}(A [\![\ldots Irr(u) \ldots]\!]) \setminus Gr_R^{\Sigma+}(A [\![\ldots (Irr(u) \wedge \neg\psi) \ldots]\!])$ is R-redundant in E. Suppose $A\sigma \in M$. This implies that $\sigma \in Sol_R^{\Sigma+}(\psi)$ and $\sigma \in Sol_R^{\Sigma+}(Irr(u))$. But then $\sigma \in Sol_R^{\Sigma+}(\varphi_i \wedge (u_j = s_i))$ for some i and j. Therefore, $\sigma \in Sol_R^{\Sigma+}(\varphi_i)$. Also $\sigma \in Sol_R^{\Sigma+}(u_j = s_i)$, so $u\sigma = s_i\sigma$. Therefore $s_i\sigma \to t_i\sigma \in Gr_R^{\Sigma+}(s_i \to t_i [\![\varphi_i]\!])$. If R does not make $s_i\sigma \to t_i\sigma$ true, then $A\sigma$ is R-redundant, since anything larger than $s_i\sigma \to t_i\sigma$ in $Gr_R^{\Sigma+}(E)$ will be R-redundant, and $u\sigma$ is smaller than some term in $A\sigma$. If R does make $s_i\sigma \to t_i\sigma$ true, then $\sigma \in Sol_R^{\Sigma+}(\neg Irr(u))$, a contradiction. □

From this we can formulate the blocking rule.

Definition 14 *Let E be a set of equations containing $A[\![\varphi[Irr(w[s'])]]\!]$ and $s \to t[\![\psi]\!]$*[6] *, such that $\sigma = mgu(s, s')$ and $\varphi\sigma$ implies $\psi\sigma$. Then the blocking rule is the constraint modification $A[\![\varphi[Irr(w[s'])]]\!] \Rightarrow [\![\varphi[Irr(w[s']) \wedge \neg\sigma]]\!]$.*

The blocking rule is most interesting when $s\sigma = s'$, so that the constraint $Irr(w[s'])$ is equivalent to $\perp$ since w can be reduced. In that case the constraint modification is $A[\![\cdots Irr(w[s'])\cdots]\!] \Rightarrow [\![\cdots\perp\cdots]\!]$. This is useful when $\psi = \top$, because then obviously $\varphi\sigma$ implies $\psi\sigma$. The previous lemma shows that the blocking rule in addition to being a constraint modification, moreover does not change the ground instances of $A[\![\varphi]\!]$ for any R. Therefore, blocking should always be performed when it is applicable, because it simplifies the constraint.

Some algorithms have been shown to be complete for quantified first order constraints with equational and disequational predicates over an extended signature (see [13, 14]).[7] Given such an algorithm, we could apply it to the modified constraint. The algorithm would ignore the irreducibility constraints. If an irreducibility predicate appears positively it is changed to $\top$. If an irreducibility predicate appears negatively, it is changed to $\perp$. This will give us all the instances that appear in any R if the set of equations that are smaller than $A[\![\varphi]\!]$ is canonical. In general, this will not be the case, so the constraint solver will not find all the redundant instances with this method. For example, suppose we have the ordering $f \succ a \succ b \succ c$ and $E = \{A[\![Irr(f(b))]\!], a \to b, a \to c\}$. Then $A[\![Irr(f(b))]\!]$ is redundant in E but the constraint solver described above will not determine that fact. That is because the set of equations smaller than A is not canonical. In fact, the only way to write a complete constraint solver is to be able to determine whether a term is irreducible in the canonical set of equations implied by all smaller equations, which is undecidable. Therefore the problem of solving our constraints is undecidable. So in practice, it is necessary to just detect reducibility in the current system.

We would like to investigate the class of unsatisfiable constraints which can be detected by an efficient algorithm. For instance, the constraints generated by (hereditary versions of) the critical pair criteria discussed in subsection 4.5 have a particularly simple form, a conjunction of simple constraints. The Basic Completion approach (and the Blocking rule above) produces a conjunction of irreducibility predicates, which can be checked for unsatisfiability by solving each one separately. The criterion of subsumed critical pairs generates constraints of the form $\neg\sigma_1 \wedge \neg\sigma_2 \wedge \ldots \wedge \neg\sigma_n$ which can be checked for unsatisfiability in the same way, by solving each of them separately. Therefore, the combination of the two types of constraints may be solved in the same way.

The point here is that we have provided a very general framework for preserving redundancy information during completion, and the techniques for encoding various existing critical pair criteria require relatively simple constraint solving techniques. The subject of more sophisticated (or simply more efficient) constraint solvers for our class of constraints is a subject of future research.

6 Completeness

We now consider the completeness of the rules presented in section 4. 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. The differences here have to do with the nature of constrained inference rules, which add and delete certain ground instances of the equations involved in a subtle way.

Definition 15 *A sequence $< S_0, S_1, \ldots >$ of sets of equations over Σ is a* derivation *from S if $S_0 = S$ and for each $i \geq 0$, for any R and Σ^+, $Gr_R^{\Sigma^+}(S_{i+1}) = (Gr_R^{\Sigma^+}(S_i) \cup E_1) \setminus E_2$ where E_1 and*

[6]This could be stated even more generally, by requiring only that $s\sigma \succ t\sigma$.

[7]For related results on solving ordering constraints, see [6, 9].

E_2 are sets of equations such that $Gr^{\Sigma^+}(erasure(S_i)) \models E_1$ and each equation in E_2 is R-redundant in S_i.

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. *A set S is* saturated *if, for every pair of equation in S, if an instance of C-Deduce exists involving those equations then some instance of C-Deduce involving those equations is redundant in S. A derivation is* fair *if the limit is saturated.*

Fair derivations can be constructed by performing all inferences among persisting equations in some systematic fashion such as breadth-first search, or breadth-first modulo size.[8] An inference rule can be viewed as a transformation from one set of equations to another set of equations. Some instances are added to the set and some deleted. The next result shows that redundant instances of equations and inferences stay redundant when equations are added or when redundant instances are deleted.

Lemma 8 *Suppose E is a set of equations over a signature Σ, and R a rewrite system. Further suppose E' is a set of equations and Σ^+ an extension such that $Gr_R^{\Sigma^+}(E) \subseteq Gr_R^{\Sigma^+}(E')$. Then:*

1. *Any equation (or inference) which is R-redundant in E is also R-redundant in E'; and*
2. *If all ground instances in $Gr_R^{\Sigma^+}(E') \setminus Gr_R^{\Sigma^+}(E)$ are R-redundant in E', then any equation which is R-redundant in E' below B is also R-redundant in E below B, and any inference which is R-redundant in E' is also R-redundant in E.*

Proof. Without loss of generality we prove the result for only ground instances of both equations and inferences. The proof for (1) is trivial, since adding more instances to $Gr_R^{\Sigma^+}(E)$ does not affect the ability to select the appropriate A_i. For (2), suppose A is R-redundant below B in E'. Choose the set $\{A_1, \ldots, A_n\}$ minimal wrt the multiset extension of the equation ordering. Suppose some A_i is not in $Gr_R^{\Sigma^+}(E)$, and thus is R-redundant. Then there is a set of equations $\{B_1, \cdots, B_m\}$ which are all smaller than A_i and such that if $R \models B_k$ for each k, then $R \models A_i$. Thus $\{A_1, \cdots, A_{i-1}, B_1, \cdots, B_m, A_{i+1}, \cdots, A_n\}$ is smaller than the original set and sufficient to prove the R-redundancy of A below B, a contradiction. This shows that each of the A_i must have been in $Gr_R^{\Sigma^+}(E)$. The case of inferences is a trivial extension of this result. □

This will suffice to show that the inference systems presented are sufficient to saturate a set of equations. We now show that saturated sets are canonical. In our framework, this will allow us to argue that our constrained completion systems will produce canonical sets in the limit. Our proof follows very much in the lines of the proof in [5], with the addition of the constraint formalism. In addition, there are some delicate features of the proof which relate to the use of an arbitrary extended signature (to play the role ordinarily played by Skolem constants) and also irreducibility constraints defined relative to a rewrite system which is constructed inductively from the set of constrained equations itself. (This induction is the reason for the use of $R_{s\sigma\approx s\sigma}$ in case (3) of Definition 2.)

First we give a method for constructing a canonical set of ground rewrite rules from a given set of equations.

Definition 16 *Let E be a set of equations over a signature Σ and $\mathcal{EQ}$ denote the set of all ground equations over some extension Σ^+. We define the ground rewriting system R_E over Σ^+ 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_A be defined as $\bigcup_{B\prec A} R_{B^+}$. Then $R_{A^+} = \{A\} \cup R_A$ if A is a member of $Gr_{R_A}^{\Sigma^+}(E)$ in the form $s \approx t$ where $s \succ t$ and s is irreducible by R_A; otherwise $R_{A^+} = R_A$. Finally define R_E as $\bigcup_{A\in\mathcal{EQ}} R_{A^+}$.*

[8]Typically completion procedures order their queues by size for efficiency.

The properties of the preceding definition we shall need are as follows.

Lemma 9 *For R_E as just defined,*

1. *If $A \in Gr^{\Sigma+}_{R_A}(E)$, then $A \in Gr^{\Sigma+}_{R_E}(E)$;*
2. *R_E and R_A for any $A \in \mathcal{EQ}$ are ground canonical;*
3. *There is no equation $A \in R_E$ such that $R_A \models A$; and*
4. *For all equations $A \in Gr^{\Sigma+}_{R_E}(E)$, $R_E \models A$ iff $R_{A^+} \models A$.*

Proof. The first claim follows from our assumption in section 2.2 that in equations of the form $s \approx t[\![\cdots Irr(u) \cdots]\!]$, $u \prec s$. For the second, clearly the fact that R_E (and any subset thereof) is terminating and left–reduced implies that it is canonical. For (3), if $R_{(s \approx t)} \models (s \approx t)$, then there would be a rewrite proof between s and t, and since $s \succ t$, then s would be reducible by $R_{(s \approx t)}$, a contradiction. For the last claim, the *if* direction, and the *only if* direction in the case of an identity are trivial. Now suppose A is in the form $s \approx t$, with $s \succ t$, and there exists a rewrite proof for A in R_E. Then s is reducible by R_E; but it is reducible at the root only if R_A includes $\{s \approx t\}$ and below the root only by a strictly smaller equation. Thus A has a rewrite proof in R_{A^+}. □

We now explore the conditions under which such a construction results in a rewrite system equivalent to the original E in a certain sense.

Theorem 1 *Let E be a saturated set of equations such that for each $A[\![\varphi_1, \varphi_2, M]\!] \in E$, φ_1 is stronger than $Irr(Var(A))$. Then R_E is equivalent to $Gr^{\Sigma+}_{R_E}(E)$.*

Proof. For convenience of notation, let R denote the set R_E. Since $R \subseteq Gr^{\Sigma+}_R(E)$, we proceed to show by contradiction that R makes true every member of $Gr^{\Sigma+}_R(E)$. Let $s \approx t$ (with $s \succ t$) be the least (wrt $\succ_{mul}$) member of $Gr^{\Sigma+}_R(E)$ such that $R \not\models s \approx t$. If s is irreducible by $R_{s \approx t}$, then by the construction, $s \approx t$ would be in R, a contradiction. Therefore s is in the form $s[l]_p$ for $l \approx r \in R$, where $s \approx t \succ_{mul} l \approx r$ and $l \succ r$. We assume that the position p is the least such reducible position wrt $<_r$. We thus have a Ground-Deduce inference

$$\frac{l \approx r \quad s[l]_p \approx t}{s[r] \approx t}$$

We now show that this inference is R-redundant, by proving that it is a ground instance of a legal C-Deduce inference on constrained equations in E.

Now we know that $l \approx r$ must be a member of some

$$Gr^{\Sigma+}_R(l' \approx r'[\![\varphi_1, \varphi_2, M]\!])$$

via a ground substitution σ_1, and $s[l] \approx t$ must be a member of some $Gr^{\Sigma+}_R(s'[u] \approx t'[\![\psi_1, \psi_2, N]\!])$ via a ground substitution σ_2. In other words, σ_1 is an R-solution of φ_1, and σ_2 of ψ_1. Since we may assume the two equations are variable–disjoint, we may form the substitution $\theta = \sigma_1 \cup \sigma_2$. Therefore there exists an inference

$$\frac{l' \approx r'[\![\varphi_1, \varphi_2, M]\!] \quad s'[u]_p \approx t'[\![\psi_1, \psi_2, N]\!]}{s'[r']\sigma \approx t'\sigma[\![\Delta_1, \Delta_2, P]\!]}$$

for $\sigma = mgu(l', u)$ and there exists η such that $\sigma\eta = \theta$. This is a legal C-Deduce inference, since u must be a non-variable term occurring in s, as θ is an R-solution of ψ_1, which is no weaker than $Irr(Var(s'[u]))$ (this implicitly uses part (1) of the previous lemma).

Now θ is a solution of $\varphi_1 \wedge \psi_1$, and $l \approx r \in R$ (so l is R-irreducible by smaller equations), and furthermore the redex position is minimal in $<_r$; thus η is an R-solution of $\varphi_1\sigma \wedge \psi_1\sigma \wedge Irr(T)$ (with T as in the definition of C-Deduce).

Clearly the ground inference is an R-instance of the C-Deduce inference. Since E is saturated, the inference, and therefore its ground instance, are R-redundant. Now, clearly $l \approx r$ is not R-redundant, by part (3) of the previous lemma. Similarly, since $s \approx t$ is the least member of $Gr_R^{\Sigma^+}(E)$ false in R, it can not be R-redundant.

Thus the only possibility which remains is that the conclusion $s[r] \approx t$ is R-redundant below $s[l] \approx t$. But then, since $l \approx r$ is also true in $R_{s \approx t}$, and $\{l \approx r, s[r] \approx t\} \models s[l] \approx t$, then $s[l] \approx t$ would be R-redundant, a contradiction.

Thus R makes true every member of $Gr_R^{\Sigma^+}(E)$. $\Box$

We now state the main completeness result of the paper.

Theorem 2 *Let E be a set of unconstrained equations and S be a set of equations of the form $A[\![\, Irr(Var(A)), \top \,]\!]$ for $A \in E$. Let $< S, \cdots >$ be a fair derivation from S. Then $Gr_{R_{S_\infty}}^{\Sigma^+}(S_\infty)$ is ground canonical and equivalent to E over ground terms. In addition the set of orientable instances of the erasure of S_∞ is a canonical rewriting system equivalent to E.*

Proof. For simplicity, let $R = Gr_{R_{S_\infty}}^{\Sigma^+}(S_\infty)$. Since S_∞ is saturated, the previous two results show that R_{S_∞} is ground canonical and equivalent to R. By Lemma 8, R is equivalent to $Gr_{R_{S_\infty}}^{\Sigma^+}(E)$, which is identical with $Gr_\emptyset(E)$ (since E has no constraints). Thus R_{S_∞} is ground canonical and equivalent to E. Since $R_{S_\infty} \subseteq R$, and by soundness of C-Deduce, S_∞ is ground canonical and equivalent to E. Finally, $R \equiv erasure(S_\infty)$ and $R \subseteq Gr^{\Sigma^+}(erasure(S_\infty))$ imply that $erasure(S_\infty)$ is ground canonical and equivalent to E on ground terms. Since we have shown this for any extension Σ^+ of our given signature Σ, $erasure(S_\infty)$ is canonical. $\Box$

The use of the arbitrary extension Σ^+ may now be seen as a substitute for the standard technique of adding Skolem constants in order to prove that the result of an ordered completion process is not only ground canonical, but canonical (cf. theorem 3 in section 6 of [5]).

This proves completeness in the ordered completion case. However, to relate this to standard completion (where failure due to unorientable equations is a possibility) and ordinary notions of (unconstrained) rewriting, the first two conditions of the C-Deduce inference rule must be replaced by $s \succ t$ and $u[s'] \succ v$.

Corollary 1 *Let E and S be as above, and let C-Deduce be modified as just mentioned. Let $< S, \cdots >$ be a fair derivation from S, where for every equation $s \approx t[\![\, \varphi \,]\!]$ in S_∞ we have $s \succ t$ or $t \succ s$. Then $erasure(S_\infty)$ is canonical and equivalent to E.*

This shows that our inference system produces a canonical rewriting system in the limit if no failure due to unorientable equations occurs. Note that the result of the process is a constrained set of rewrite rules whose constraints do not alter the equational theory of the set of rules or the fact of its being canonical, but only remove some unnecessary instances of the rules. However, if solving such constraints during the rewriting process is undesirable (e.g., if they are undecidable), then our results also show us that the constraints can simply be erased.

7 The Recalculation Problem

This paper is about methods for eliminating unnecessary computation in completion procedures. An interesting problem along these lines is one we call the *recalculation problem*. Typically in a completion procedure a subset SAT of equations is maintained which are saturated (i.e., all critical pairs among the equations have already been generated). After equations are oriented and used for simplification, they may be used to generate critical pairs and then added to SAT. The problem is that when equationss in this set are simplified on the bigger side, they need to be re-oriented and to go through the saturation process with the other equations in SAT again, potentially reconstructing critical pairs already in the system. (It can be shown using the techniques of redundancy developed in [3], and hence in our notions of redundancy, that simplification of the smaller side of an equation in SAT does not require such recalculation, since any newly calculated critical pairs would be redundant.)

In general, the notion of a fair derivation sequence does not provide enough information to give us interesting criteria for eliminating such recalculation. It would be useful to know what additional properties a completion procedure must have to avoid certain kinds of recalculation.

Our framework allows us to show precisely which critical pairs need to be recalculated when the left hand side of a equation is simplified. Suppose that an equation $s \approx t[\![\varphi]\!] \in SAT$ (with $s \succ t$) is simplified at position p into $s' \approx t[\![\varphi']\!]$ (retaining its orientation of $s' \succ t$). Then the question is which overlaps of the new equation with the other equations of SAT need to be considered for completeness. Our results show that in any system where the constraints are never weakened, the right premise is strengthened as much as possible, and the constraints are inherited by the conclusion of an inference, only overlaps strictly above p need be considered.

In fact we can show a more general result concerning not just simplification steps, but arbitrary inference steps, because the system cannot distinguish between the two. Since every critical pair inference is considered as a simplification, the inferences do not need to be recalculated. Therefore, this is the one example we know of where restrictions imposed by constraints actually allows one to be more restrictive elsewhere. The tradeoff comes when the disequational constraint which prevents recalculation also prevents other simplifications from being done, as we show in an example below.

The following theorem assumes that the CM1 constraint modification rule is possible. Therefore the theorem holds for standard completion, because CM1 is always possible. In the case of ordered completion, the result only holds for inferences where the left premise and conclusion are smaller than the right premise. This requires that the left premise be ordered. To prove the result, we just need to show that certain inferences are redundant because the constraint is not satisfied.

Theorem 3 *Let I be an instance of C-Deduce of the form*

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

where Δ_1 is no weaker than $\varphi_1\sigma \wedge \psi_1\sigma$ and ψ_1 is modified to a constraint that is no weaker than $\psi_1 \wedge \neg(\sigma \wedge \varphi_1)$. Then the completion procedure is complete for inference rule I, with the restriction that after the above inference is performed, no further inference of $s \approx t$ needs to be performed into $u[s'] \approx v$ at position p. Also no inference of $s \approx t$ needs to be performed into any later descendant[9] of $u[s'] \approx v$ at position p, unless an inference is performed at a position above, at or below p to create that descendant.

Proof. We must show that after an inference is performed at position p, the constraint disallows that inference to be repeated, that constraints never become weaker, and that constraints are inherited by the conclusion of an inference. The first condition is true because, after an inference is performed the constraint is modified to a constraint that is no weaker than $\psi \wedge \neg(\sigma \wedge \varphi)$. This shows that the inference does not need to be redone, because it would be redundant. The second condition is given. The third condition is a result of the fact that the constraint on the conclusion of an inference is not any weaker than $\varphi_1\sigma \wedge \psi_1\sigma$. This ensures that constraints from both premises are inherited by the conclusion of an inference. $\square$

[9]By descendant of A, we mean a conclusion of an inference with some premise A, or any descendant of the conclusion.

In other words, in these systems, overlaps at disjoint positions need not be recalculated after a critical pair inference or a simplification. The reason is that these critical pairs would involve instances which are now redundant, and hence can be ignored. An example of this type of system is CCP.

We now give an example showing that the above theorem does not hold in general for standard completion (without constraints). In other words, we exhibit a set of equations, and an order of critical pair inference and simplification steps that is seemingly fair, but a canonical rewriting system is not produced (unless critical pairs into disjoint positions of simplified terms are allowed). There is an order of critical pair inference and simplification steps which would result in a canonical inference system, however there is no way for a completion procedure to know in advance which is the best way to order the inference and simplification steps to avoid recalculations. We emphasize that this example is intended merely to show that the above theorem does not hold in general, therefore the example is as simple as possible.

Consider the following set of equations under the following ordering: $a > f > g > b > c > d > e$.

- $g(b) \rightarrow c$
- $f(a, g(x)) \rightarrow e$
- $a \rightarrow d$
- $a \rightarrow f(d, g(b))$

First form a critical pair between $f(a, g(x)) \rightarrow e$ and $g(b) \rightarrow c$. You get $f(a, c) \rightarrow e$ which simplifies by $a \rightarrow f(d, g(b))$ to $f(f(d, g(b)), c) \rightarrow e$. Then simplify $f(a, g(x)) \rightarrow e$ by $a \rightarrow d$ to $f(d, g(x)) \rightarrow e$. Simplify $f(f(d, g(b)), c) \rightarrow e$ by $f(d, g(x)) \rightarrow e$ to $f(e, c) \rightarrow e$. Simplify $a \rightarrow f(d, g(b))$ by $g(b) \rightarrow c$ and $a \rightarrow d$ to $f(d, c) \rightarrow d$.

The final rewrite system is:

- $g(b) \rightarrow c$
- $f(d, g(x)) \rightarrow e$
- $a \rightarrow d$
- $f(d, c) \rightarrow d$
- $f(e, c) \rightarrow e$

There are no more inferences that can be performed unless we recalculate critical pairs. But this is not a canonical rewrite system. For instance $d \approx e$ but they are both in normal form.

Suppose we re-do the above derivation using disequational constraints. In that case, we first form a critical pair between $f(a, g(x)) \rightarrow e$ and $g(b) \rightarrow c$ to get $f(a, c) \rightarrow e$. Then, we can add a disequational constraint to the equation $f(a, g(x)) \rightarrow e$, so it becomes $f(a, g(x)) \rightarrow e [\![x \neq b]\!]$. As before, the equation $f(a, c) \rightarrow e$ can be simplified by $a \rightarrow f(d, g(b))$ to $f(f(d, g(b)), c) \rightarrow e$. Then simplify $f(a, g(x)) \rightarrow e [\![x \neq b]\!]$ by $a \rightarrow d$ to $f(d, g(x)) \rightarrow e [\![x \neq b]\!]$. Now, because of the disequational constraint, we cannot simplify $f(f(d, g(b)), c) \rightarrow e$ by $f(d, g(x)) \rightarrow e [\![x \neq b]\!]$. The reader will note that the set of equations can now be completed. The point of this example is that the disequational constraint prevents recalculation of critical pairs, but at the same time disallows some simplifications. It is another example of the tradeoffs that must be made when deciding how constraints are used.

8 Initial Constraints

In this section, we extend the previous results to sets of equations with initial constraints. This requires us to re-define the definition of erasure and some of the previous constraint modification rules. It also forces us to effectively paramodulate below variables in some instances. The necessity of paramodulating below variables can create an inefficient theorem prover. Therefore the results of this section may not be useful in practice. However, even if that is the case, we think this section is very useful in that it presents a framework which shows explicitly which constraints are used to control the inference and which constraints are necessary to prove soundness.

Throughout this paper, we have been assuming that all equations are initially unconstrained. But what happens if the completion procedure tries to operate on a set of constrained equations? Unfortunately, completeness is lost. Consider the following example.

Suppose we have an ordering $f \succ g \succ h \succ a \succ b$ and $E = \{f(x) \approx g(x) [\![x \neq h(b)]\!], a \approx b\}$. There are no inferences among these two equations. We can show that $f(h(b)) \approx g(h(b)))$ because $f(h(b)) \approx f(h(a)) \approx g(h(a)) \approx g(h(b))$. However neither $f(h(b))$ nor $g(h(b))$ can be rewritten using equations in E. So E is not canonical.

One way to deal with this situation would be to weaken the constraint $f(x) \approx g(x) [\![x \neq h(b)]\!]$ as we sometimes need to weaken the constraint on a simplifier. Unfortunately, this is not sound. So we have to deal with the problem in some other way. What we can do is to superpose at variables or below variables. For the example above, it is not enough to superpose at variables. It is necessary to unify the variable x with $h(a)$ which contains the left hand side of a rewrite rule as a subterm, effectively using a functional reflexivity axiom. This process is described below. Another alternative, discussed in [11], is to expand out the constraint based on the signature. For example, a disequation $x \neq f(y)$ is equivalent to $x = a \vee x = b \vee (x = f(z) \wedge z \neq y)$ if the signature is simply $\{a, b, f\}$. Note that this does not work in our context of extended signatures.

To handle initial constraints we must add one more parameter to each constrained equation. We will represent a constrained equation as $A [\![\varphi_1, \varphi_2, M, \varphi_3]\!]$. The first three parameters represent the same thing as before. The fourth parameter has been added to represent all of the instances that are true by equations less than or equal to M. For all R, if $B \in Gr_R(A [\![\varphi_3]\!])$ then B is redundant in E up to M. We define $erasure(A [\![\varphi_1, \varphi_2, M, \varphi_3]\!]) = A [\![\varphi_3]\!]$. This new definition of erasure is necessary to ensure the soundness of the completion procedure.

The constraint modification $A [\![\varphi]\!] \Rightarrow A [\![IrrVar(A, \varphi), NoIrrVar(A, \varphi)]\!]$ was discussed in section 3. We gave an example of the result of this constraint modification on a constrained equation. For an unconstrained equation the result of $IrrVar(A, \varphi)$ is a conjunction of irreducibility conditions on all the variables in A. These constraints are passed on in inferences, so they enforce the restriction that no superposition take place at or below a variable. However, for constrained equations, the result of $IrrVar(A, \varphi)$ is not a conjunction on all the variables, so we are not forbidden to superpose at or below variable positions. As we saw in the example above, this is necessary for completeness. With the additional constraint, the C-Deduce rule is as follows.

C-Deduce

$$\frac{s \approx t [\![\varphi_1, \varphi_2, M, \varphi_3]\!] \qquad u[s'] \approx v [\![\psi_1, \psi_2, N, \psi_3]\!]}{u[t]\sigma \approx v\sigma [\![\Delta_1, \Delta_2, M\sigma \cup N\sigma, \varphi_3\sigma \wedge \psi_3\sigma]\!]}$$

The only difference between this definition and the original definition is the fourth parameter of the constraint. The conclusion inherited the conjunction of the constraints of the premises with σ applied. Those are the only instances of the conclusion that we know are true in the theory. This parameter is very important to preserve soundness. We must use this parameter if we want to add new instances of an equation by changing the first parameter of the constraint.

However, this rule is not good enough to preserve completeness. We need to add an additional version of the C-Deduce rule that allows us to superpose below variables. The inference looks like this:

C-Deduce-Below

$$\frac{s \approx t\,[\![\, \varphi_1, \varphi_2, M, \varphi_3 \,]\!] \qquad u[x] \approx v\,[\![\, \psi_1, \psi_2, N, \psi_3 \,]\!]}{u[w[t]]\sigma \approx v\sigma\,[\![\, \Delta_1, \Delta_2, M\sigma \cup N\sigma, \varphi_3\sigma \wedge \psi_3\sigma \,]\!]}$$

where w is some arbitrary term containing s.

The use of w is necessary because we could have unified x with any term w containing s as a subterm. So $\sigma = [x \mapsto w[s]]$. This inference is redundant if $\psi_1\sigma$ is not true. Moreover, the inference is not sound if $\psi_3\sigma$ is not true. So we must not perform the inference in that case. The inference is also redundant if $\psi_2\sigma$ is true, because then the conclusion of the inference follows from the right premise. Since we are superposing into variables, this procedure may be useless in practice. However, the redundancy criteria may restrict the inference enough so it is useful.

When dealing with initially constrained equations, we have to be careful not to add instances of equations which are not implied by the theory. That is what the fourth parameter of the constraint is used for. For example, in section 4 we gave some correct constraint modification rules and then gave particular examples of those rules. Some of the examples required adding instances of the left premise or conclusion in order to simplify more instances of the right premise. In order not to lose soundness we have to alter the modifications of the constraints that add instances.

We need to change the constraint modifications as follows. For the C-Simplify rule Δ_1 should be set equal to $\psi_1\sigma \wedge \varphi_3\sigma$ instead of $\psi_1\sigma$. This is because we can only add instances that we know follow from the theory. The constraint modifications are replaced by the following: $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge \varphi_3)$, $\varphi_1 \Rightarrow \varphi_1 \vee (\sigma \wedge \psi_1 \wedge \varphi_2 \wedge \varphi_3)$, $\varphi_2 \Rightarrow \varphi_2 \vee (\sigma \wedge \psi_1 \wedge \varphi_3)$.

A similar thing happens in the CCP1 and CCP2 inferences. For CCP1, Δ_1 is set equal to $\varphi_2\sigma \wedge \psi_1\sigma \wedge Irr(T) \wedge \varphi_3\sigma$. For CCP2, Δ_1 is set equal to $\psi_1\sigma \wedge Irr(T) \wedge \varphi_3\sigma$ and the constraint modification on ψ_1 becomes $\psi_1 \Rightarrow \psi_1 \wedge \neg(\sigma \wedge Irr(T) \wedge \varphi_3)$.

For initially constrained equations, the constraint solver must be complete for the kind of constraints that may appear as the fourth parameter (i.e., the initial constraints). Otherwise, the system is not sound.

Given the changes in the definition of erasure and constraint modification to preserve soundness, and the change to the inference rule to preserve completeness, the completeness proof of section 6 applies directly to the case of initial constraints. Note that some of the definitions have been presented in a general way, so that they would apply to the case of initially constrained equations.

We must remember the remark given in section 2.2 of this paper. The remark stated that irreducibility constraints will always be kept in such a form that the term in the irreducibility constraint is smaller than the equation. That causes a problem for initial irreducibility constraints, because, in order to preserve completeness we must keep the irreducibility constraints in this form. However, in order to preserve soundness we are not allowed to weaken constraints that derive from initial constraints. This is a contradiction, so we in general cannot handle initial irreducibility constraints. For example, the set

$$a \approx b\,[\![\, Irr(b) \,]\!],\ a \approx c\,[\![\, Irr(b) \,]\!],\ b \approx e\,[\![\, Irr(a) \,]\!]$$

where $a \succ b \succ c \succ e$, has no meaning, according to our semantics, and we can not perform the overlap of the first two equations. The difficulties with initial irreducibility constraints is unfortunate, and perhaps there is some restricted form in which they are tractable. This would be useful in that initial irreducibility constraints can be used to semantically constrain variables. For example, in a field we would like to have an axiom $x * x^{-1} = 1\,[\![\, x \neq 0 \,]\!]$ but include the fact that x is not only syntactically distinct from 0, but also is not equivalent to 0 in the theory. This can be represented by the axiom $x * x^{-1} = 1\,[\![\, x \neq 0 \wedge Irr(x) \,]\!]$ since in general 0 will be irreducible and so the irreducibility constraint on x embeds the condition that the binding for x not be equivalent to 0.

However, initial equational and disequational constraints may be handled as discussed above. For that case, we have described a sound and complete inference procedure, i.e., we need the two rules C-Deduce and C-Deduce-Below. The constraint modifications still apply, but we must ensure that the left premise is never weakened more than φ_3.

9 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 rigorous 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 [5] (see also [15]), which in turn adapted the results of [1] (cf. [17] and [25]). However, the inference systems are developments of the rules from the seminal paper [11] to show how irreducibility constraints can be used to express the idea of Basic Completion in combination with other kinds of equational constraints to encode other critical pair critiria such as subsumed critical pairs. Nieuwenhuis and Rubio [15] also expressed Basic Completion in terms of constraints. However, they used equational constraints instead of irreducibility constraints. We prefer to use irreducibility constraints, so as not to confuse them with equational constraints.

The completion system in [11] 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 section 8, completion is not complete with initially constrained equations unless we allow superposing into variables. In order to insure completeness [11] considered some additional inference rules which basically had the purpose of turning constrained equations into unconstrained equations. In section 8 we showed how completeness can be preserved with initial constraints by allowing a limited form of variable overlap. Our completeness proof is the first one we are aware of for equational and disequational constraints without any additional rules, except for the extension of C-Deduce. Also Nieuwenhuis and Rubio [16] have a method of dealing with initial ordering constraints without adding any additional rules, and equational and disequational constraints can be encoded by ordering constraints. However, they only allow a specific class of initial constraints, while we allow all initial constraints at the cost of the extension of C-Deduce. We should remark that although we do not consider ordering constraints, it seems that they could be added to our system without major alterations of the framework. This topic is left for future research.

We do not expect that this framework in its entirety would be necessarily be an efficient and usable 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 focuses 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.

10 Acknowledgements

We would like to thank Claude Kirchner for his encouragement and we would also like to thank the anonymous referees for pointing out numerous errors and providing suggestions for constructive changes. Also, we thank Hubert Comon for his remarks on constraint solving in extended signatures.

References

[1] L. Bachmair and H. Ganzinger. Rewrite-based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation* **4** (1993) pp. 1–31.

[2] L. Bachmair and N. Dershowitz. Critical Pair Criteria for Completion. *Journal of 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. *Proc. 11th Conference on Automated Deduction*, Springer-Verlag LNAI **607**, Berlin (1992) pp. 462–476.

[5] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation and Superposition. To appear in *Journal of Information and Computation*.

[6] H. Comon. Solving Symbolic Ordering Constraints. *International Journal of Foundations of Computer Science* **1** (1990) pp. 387–411.

[7] J. Hsiang and M. Rusinowitch. Proving Refutational Completeness of Theorem Proving Strategies: The Transfinite Semantic Tree Method. *Journal of the ACM* **38** (1991) pp. 559–587.

[8] G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the ACM* **27** (1980) pp. 797–821.

[9] J-P. Jouannaud and M. Okada. Satisfiability of Systems of Ordinal Notations with the Subterm Property is Decidable. *Automata Languages and Programming, 18th International Colloquium*, Springer-Verlag LNCS **510**, Berlin (1991).

[10] D. Kapur, D. Musser and P. Narendran. Only Prime Superpositions Need to be Considered in the Knuth-Bendix Completion Procedure. *Journal of Symbolic Computation 6* (1988) pp.19–36.

[11] C. Kirchner, H. Kirchner and M. Rusinowitch. Deduction with Symbolic Constraints. *Revue Francaise d'Intelligence Artificielle* **4:3**, (1990) pp. 9–52.

[12] W. Küchlin. A Confluence Criterion Based on the Generalized Newman Lemma. *Proc. Eurocal '85*, Springer-Verlag LNCS **204**, Berlin (1985) pp.390–399.

[13] K. Kunen. Answer Sets and Negation as Failure. *Proc. 4th International Conference on Logic Programming*, MIT Press, Cambridge MA (1987).

[14] M. Maher. Complete Axiomatization of the Algebras of Finite, Rational and Infinite Trees. *Proc. 3rd Annual Symposium on Logic in Computer Science*, IEEE COmputer Society Press, Los Alimitos CA (1988) pp. 348–357.

[15] R. Nieuwenhuis and A. Rubio. Basic Superposition is Complete. *Proc. European Symposium on Programming*, Springer-Verlag LNCS **582**, Berlin (1992) pp. 371–389.

[16] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering Constrained Clauses. *Proc. 11th Conference on Automated Deduction*, Springer-Verlag LNAI **607**, Berlin (1992) pp. 477–491.

[17] J. Pais and G. Peterson. Using Forcing to Prove Completeness of Resolution and Paramodulation. *Journal of Symbolic Computation* **11** (1991) pp.3–19.

[18] G. Peterson. Complete Sets of Reductions with Constraints. *Proc. 10th Conference on Automated Deduction*, Springer Verlag LNCS 449, Berlin (1990) pp. 381–395.

[19] G. Smolka. *Logic Programming over Polymorphically Order-Sorted Types*. PhD thesis, Universität Kaiserslautern, FB Informatik, West-Germany (1989).

[20] W. Snyder. *A Proof Theory for General Unification*. Birkhauser Boston Inc., Boston MA (1991).

[21] F. Winkler. Reducing the Complexity of the Knuth-Bendix Completion Algorithm: A Unification of Different Approaches. *Proc. Eurocal '85*, Springer Verlag LNCS **204**, Berlin (1985) pp.378–389.

[22] F. Winkler and B. Büchberger. A Criterion for Eliminating Unnecessary Reductions in the Knuth-Bendix Algorithm. *Proc. Coll. on Algebra, Combinatorics and Logic in Computer Science*, Gyor Hungary, pp. 849–869.

[23] H. Zhang and D. Kapur. Consider only General Superposition in Completion Procedures *Proc. 3rd International Conference on Rewriting Techniques and Applications*, Spring-Verlag LNCS **355**, Berlin (1989) pp.513–529.

[24] H. Zhang and D. Kapur. *Unnecessary Inferences in Associative-Commutative Completion Procedures Mathematical Systems Theory* **23** (1990) pp. 175–206.

[25] H. Zhang. *Reduction, Superposition, and Induction: Automated Reasoning in an Equational Logic*. Ph.D. Thesis, Rensselaer Polytechnic Institute (1988).