Proof Lengths for Equational Completion [*]

David A. Plaisted

Department of Computer Science

University of North Carolina at Chapel Hill

Chapel Hill, NC 27599-3175

e-mail: plaisted@cs.unc.edu

Andrea Sattler-Klein

Universität Kaiserslautern

Fachbereich Informatik

D-67663 Kaiserslautern

e-mail: sattler@informatik.uni-kl.de

SR–95–06

Abstract

We first show that ground term-rewriting systems can be completed in a polynomial number of rewriting steps, if the appropriate data structure for terms is used. We then apply this result to study the lengths of critical pair proofs in non-ground systems, and obtain bounds on the lengths of critical pair proofs in the non-ground case. We show how these bounds depend on the types of inference steps that are allowed in the proofs.

1 Introduction

We are interested in developing theoretical techniques for evaluating the efficiency of automated inference methods. This includes bounding proof sizes, as well as bounding the size of the total search space generated. Such investigations can provide insights into the comparative strengths of various inference systems, insights that might otherwise be missed. This can also aid in the development of new methods and new inference rules, as we will show.

We first consider equational deduction for systems of ground equations. We note that in general, a system of ground equations can be converted to a ground term-rewriting

[*]This research was partially supported by the National Science Foundation under grant CCR-9108904

system by orienting the equations relative to a total termination ordering. Completion of such term-rewriting systems is basic to equational deduction, since we can test if a set E of equations logically implies an equation $s = t$ by regarding E as a term-rewriting system and completing $E \cup \{s = a, t = b\}$, where a and b are small new constant symbols, and testing whether the equation $a = b$ is in the completed system. We present two polynomial time methods for completing a ground term-rewriting system. This is significant because of the wide use of critical-pair methods in theorem proving and term-rewriting. Since an already-completed version of a ground system can be found in polynomial time using congruence closure, one would expect that completion itself by rewriting operations should be possible in polynomial time for ground systems. However, the proof of this turned out to be surprisingly difficult to find, though the final algorithms are reasonably simple. It is also surprising that no polynomial time method for critical pair completion of ground systems has been found until now, because this is such a basic problem in term-rewriting systems. Furthermore, the algorithms with polynomial behavior seem to have some unexpected features and implications for the efficiency of ground completion. One interesting feature of these algorithms is that a certain kind of data structure for terms needs to be used to obtain the polynomial behavior. We are not aware of any work prior to this which derives any time bound for ground completion, except for methods based on congruence closure.

We then apply these results on ground systems to obtain bounds on proof length for non-ground equational systems. Undecidability considerations make this more difficult, but we obtain some results in spite of this. In particular, we derive bounds on the number of steps needed in general to derive equational consequences of non-ground (first-order) equational systems using completion. This is possible because "unfailing completion" [BDP89] is a complete theorem proving method for first-order equational systems, even those that cannot be completed in a finite number of steps. We show how the derived bounds on proof length depend on the operations used in completion, and show that non-standard operations are needed to obtain good bounds. Furthermore, a certain kind of data structure for terms is needed to obtain these bounds. We also give a special case in which a better bound is satisfied. Along the way, some advantages of the rigid (matings) approach to theorem proving are revealed. In addition, we prove some properties of the needed ground instances, showing that "small" instances always exist, in a certain sense. It is interesting in this regard that [Lyn95] has recently given an extension of congruence closure to the first-order case, which may turn out to be useful for deductive purposes.

We consider that these results are important for a theoretical understanding of the efficiency of term-rewriting based theorem provers. As in first-order theorem proving, there have been many studies of the correctness or completeness of term-rewriting deductive systems, but little theoretical study of their efficiency. Such a study is invaluable for gaining a deeper machine-independent insight into the behavior of term-rewriting inference, and helping to develop more efficient inference strategies. The results presented here are a beginning in this endeavor, and help to prepare the way for a more thorough study of the efficiencies of various approaches to the non-ground case. For previous work in the length of derivations in string-rewriting systems, see

[BO84], where it is shown that systems exist whose word problem is decidable, but deciding the word problem by any canonical system can be arbitrarily more complex than deciding it by a Turing machine. The lengths of proofs in non-canonical systems in this same framework was studied in [MO85]. For a more recent paper on the same topic, see [CMO93]. Our work differs from these in that we consider proofs involving completion steps, and we relate the length of the proof to the size of an "amplification" needed to obtain the proof, rather than to the complexity of the word problem per se. Thus our results are concerned with the relationship between the complexity of the proof and the number of instances of each equation that are needed to obtain the proof. The existence of small canonical systems that take a very long time to obtain by completion was shown in [MSKO93]. Such systems would require a very large amplification, and so our bounds would still apply to them.

We begin with some definitions. A term is said to be a *ground term* if it contains no variables; thus, $f(g(a), b)$ is a ground term. We can also speak of ground equations, et cetera. We use the equivalence relation $\equiv$ for the identity relation on terms, and also for logical equivalence of first-order formulas. A *term-rewriting system* R is a finite set $\{r_i \rightarrow s_i : i \in I\}$ of rules, where r_i and s_i are terms and every variable in s_i must also appear in r_i. For surveys of term-rewriting, see [DJ90, Pla93, Klo92]. A *substitution* is a mapping from variables to terms, extended to terms and clauses homomorphically. We assume all substitutions are the identity on all but finitely many variables. We write $\{x_1 \leftarrow t_1, \cdots, x_n \leftarrow t_n\}$ for the substitution replacing the variables x_i by the terms t_i, respectively. If Θ is a substitution and t is a term, then we call $t\Theta$ an *instance* of t; similar terminology applies to instances of equations, rewrite rules, and clauses. We use the notation $t[w]$ to indicate one occurrence (or, sometimes, all occurrences) of the subterm w in the term t. We define the rewrite relation $\rightarrow_R$ on general terms by $t[r_i\Theta] \rightarrow_R t[s_i\Theta]$ where Θ is a substitution; that is, instances of r_i may be replaced by the corresponding instances of s_i. The reflexive transitive closure of this relation is indicated by $\rightarrow_R^*$. A term t is *reducible* if there is a term u such that $t \rightarrow_R u$; otherwise, t is *irreducible*. We say a term u is a *normal form* of t if $t \rightarrow_R^* u$ and u is irreducible. We say R is *terminating* if there are no infinite sequences $t_0 \rightarrow_R t_1 \rightarrow_R t_2 \ldots$ and R is *confluent* if for all terms t, t_1, and t_2, if $t \rightarrow_R^* t_1$ and $t \rightarrow_R^* t_2$ then there exists a term u such that $t_1 \rightarrow_R^* u$ and $t_2 \rightarrow_R^* u$. We say R is *convergent* or *canonical* if R is terminating and confluent. Such systems are especially interesting, because they can be used to decide the equational theory of R. If R is the term-rewriting system $\{r_i \rightarrow s_i : i \in I\}$ then we define $R^=$ to be the set $\{r_i = s_i \ : i \in I\}$ of equations. It turns out that if R is canonical, then $R^= \models t_1 = t_2$ iff t_1 and t_2 have the same normal form with respect to R-rewriting. Thus we can use R for theorem proving in the equational theory $R^=$. If R is not canonical, we may want to *complete* it, that is, find another system R_1 such that $R^= \equiv R_1^=$ and such that R_1 is canonical; then R_1 may be used to decide the equational theory $R^=$ of R. In practice, *critical pair* approaches to completion are generally used. These methods essentially modify the rules of R incrementally in an attempt to make it confluent, while preserving termination.

Definition 1.1 *Suppose $r_1 \rightarrow s_1$ and $r_2 \rightarrow s_2$ are two rules in R. Suppose r_1 has a subterm t that unifies with r_2; thus, $r_1 \equiv r_1[t]$. Let α be a most general unifier of t and*

r_2. *Then we call the pair* $(r_1[s_2]\alpha, s_1\alpha)$ *of terms a* critical pair*; we view the equation* $r_1[s_2]\alpha = s_1\alpha$ *as being derived from the two equations* $r_1 = s_1$ *and* $r_2 = s_2$ *by one critical pair operation.*

We note that a critical pair constructed from two rules in R is a logical consequence of $R^=$. It is known that if R is not confluent, then there must be a critical pair (u_1, u_2) between two rules of R such that u_1 and u_2 have different R-normal forms u_1' and u_2'. Then the equation $u_1' = u_2'$ can often be oriented into a rewrite rule and added to R. Critical pair methods perform this operation repeatedly, attempting to complete R to an equivalent canonical system. For ground systems, if there is a critical pair between rules $r_1 \rightarrow s_1$ and $r_2 \rightarrow s_2$, then one of the left-hand sides (say, r_1) must be reducible by the other rule $r_2 \rightarrow s_2$. Then we have the critical pair (r_1', s_1), where r_1' is r_1 with this rewrite performed. This can be oriented into the rule $r_1' \rightarrow s_1$ or $s_1 \rightarrow r_1'$, depending on the ordering, and for ground systems the original rule $r_1 \rightarrow s_1$ can be deleted from the system. If R is a ground system, then we can complete it by repeatedly performing such rewrites of rules with respect to other rules, until no more reductions are possible. This must terminate if a suitable termination ordering is used for orienting rules of R and the critical pairs. Also, when no more critical pairs exist, then R is canonical. In addition, even if R is canonical, some right-hand sides of rules may be reducible with respect to other rules of R; we usually want to rewrite such right-hand sides of rules to normal form. The question arises how efficiently this completion and rewriting of R may be done. It is possible to construct examples in which an unskilful choice of rewrites can lead to an exponential time process; for example, consider rules for binary counting of the form

$$f(c) \rightarrow g(c)$$
$$f(g(c)) \rightarrow g(f(c))$$
$$f(g(g(c))) \rightarrow g(f(f(c)))$$
$$\ldots$$
$$f(g^n(c)) \rightarrow g(f^n(c))$$

Although this system is canonical, the right-hand sides can be further rewritten. The straightforward reduction of the term $g(f^n(c))$ can take a number of rewrites exponential in n. However, if we apply the rules in order of size, smallest first, to all other rules, the whole system can be rewritten to a reduced system in a polynomial number of steps. In [GNP+93, Sny89], a general, polynomial time method was presented for obtaining completed ground systems. This method was based on congruence closure, and therefore did not give direct insight into the speed of completion by traditional critical pair-based methods. The question remained whether a good choice of critical pair and rewriting operations could always complete and rewrite a ground system in polynomial time, relative to an arbitrary total termination ordering. In this paper, we give two polynomial time methods for doing this. The first method constructs a subset D of the terms appearing in the ground system R. Initially, D is equal to all the subterms appearing in rules of R. As rewrite rules are applied to R, they are also applied to D. This is done in such a way as to decrease the cardinality of D. This decrease in cardinality guarantees the polynomial running time of this method. The second method considers the set of all the subterms appearing in rules of R. Rewrites are performed in such a way as to reduce the cardinality of this set. Both methods have the unexpected feature that they give priority to rewrite rules whose right-hand sides are small.

2 Ground systems

We now discuss the general features of our first ground completion method. This method performs a sequence of rewrite operations on a ground system, and permits some nondeterministic choice in the rewrites that are performed. The rewrite rules are oriented using an arbitrary termination ordering $>$ on ground terms. This ordering must be well-founded and satisfy the monotonicity property $r > s$ implies $f(\ldots r \ldots) > f(\ldots s \ldots)$. We also assume that this ordering is total. We write $s \geq t$ to indicate $s > t$ or $s \equiv t$. Also, $s > t$ iff $t < s$, and $s \geq t$ iff $t \leq s$. Given an arbitrary ground term-rewriting system R, our algorithm constructs an equivalent canonical system S such that all rules in S are oriented with respect to $>$, that is, their left-hand sides are larger than their right-hand sides. Also, all rules of S are fully rewritten with respect to other rules in S. These two properties are sufficient to guarantee canonicity of S, that is, S is confluent and terminating. Moreover, a system S satisfying these two properties is unique, given R and the ordering $>$.

The idea of our completion method is to choose a rule $r \rightarrow s$ of R such that $r > s$ and *process* it, that is, replace all other occurrences of r by s. If $r < s$, then we need to *re-orient* this rule to $s \rightarrow r$ before processing. Note that processing this rule may cause other rules $r' \rightarrow s'$ to be created in which the left-hand side r' is smaller than the right-hand side s', that is, $r' < s'$. This can happen when r' has been rewritten, for example. In our method, we assume that such rules are immediately re-oriented to $s' \rightarrow r'$ so that the left-hand side is larger than the right-hand side. Also, if r' and s' are identical, then the rule $r' \rightarrow s'$ is simply deleted. The problem is to choose a sequence of rules $r \rightarrow s$ to process so that completion can be done in polynomial time. To do this, we choose a positive integer function $c(R)$ of R as a termination function. That is, this function initially has a (positive) value that is polynomial in the size of R, and each time a rule is processed, the value of the function $c(R)$ decreases by at least one. It follows that the completion procedure terminates in a polynomial number of processing steps. Since we re-orient rules often, we need to choose a termination measure $c(R)$ that is unaffected by this reorientation of rules. If each processing step can be done in polynomial time, then the entire completion process will require time at worst polynomial in the size of R. In order to be able to process a rule $r \rightarrow s$ in polynomial time, we need to be able to replace all occurrences of r by s quickly. There may be exponentially many occurrences of r; for example, we can have a system containing rules like $c_i \rightarrow f(c_{i+1}, c_{i+1})$. We need to be able to rewrite all the occurrences of r at the same time. For this reason, we assume that terms are represented by *directed acyclic graphs*, so that all occurrences of a given subterm are represented in one location, and all can be rewritten with an amount of work proportional to the work required to rewrite a single occurrence, and independent of the number of occurrences. Such data structures are well known. Assuming that such a directed acyclic graph representation is used for terms, each processing step is polynomial. However, if terms are represented in a conventional way, these processing steps can take an exponential number of rewrite operations. Still, we feel that the directed acyclic graph representation is natural enough so that it is justifiable to speak of this as a polynomial time completion method.

Our termination function is based on the concept of *dominating sets*. The idea is to count the number of distinct right-hand sides of rules that appear in R. Each right-hand side can be considered as a name of an equivalence class of terms that has been detected so far. However, as rules are rewritten and reoriented, the distinction between left and right-hand sides becomes somewhat arbitrary. So instead of counting the number of right-hand sides, we just choose some arbitrary set D containing for each rule $s \rightarrow t$ of R, either s or t. We also want to take into account the subterms of R that do not appear on the left or right-hand side of any rewrite rule. Therefore, for each subterm u in R, if there is no rewrite rule $s \rightarrow t$ with $u \equiv s$ or $u \equiv t$, then u is in D. The elements of D can be seen as labels of equivalence classes of terms.

Definition 2.1 *A* dominating set *D for R is a set of terms such that for every rule $s \rightarrow t$ in R, either $s \in D$ or $t \in D$. Also, for each subterm u in R, either u is in D or there is a rewrite rule $s \rightarrow t$ having u on the left or right-hand side. We assume that at the beginning D is chosen as the set of all the subterms appearing in R, and D is then updated during processing. That is, when a rule is processed, the elements of D are also rewritten, if possible. It will turn out that this processing will maintain the property that D is a dominating set. One goal of the processing will be to reduce the cardinality of D. We use $|A|$ to indicate the cardinality (number of elements) of a set A.*

Lemma 2.2 *The processing of a rule $r \rightarrow s$ always rewrites a dominating set D of R into another dominating set D' of R'.*

Proof. If $r \in D$ then $s \in D'$, so the dominating property is preserved for the rule $r \rightarrow s$. If $u \rightarrow v$ is another rule that is rewritten to $u' \rightarrow v'$ by processing, then $u' \in D'$ or $v' \in D'$, since $u \in D$ or $v \in D$. If t is a term that does not appear on the left or right-hand side of a rule of R, then $t \in D$. If t is rewritten to t' in R', then $t' \in D'$. If there is a rewrite rule $u \rightarrow v$ in R having t on the left or right-hand side, and this is rewritten to $u' \rightarrow v'$ in R', then t' is either u' or v'. If the rule $u' \rightarrow v'$ is deleted from R' because $u' \equiv v'$, then $u' \in D'$, so $t' \in D'$, too, as required for terms that do not appear in rewrite rules. □

Definition 2.3 *A rule is a* bridging rule *if it is of the form $s \rightarrow t$ where both s and t are in the dominating set D. A rule is a* linking rule *if it is of the form $s \rightarrow t$ (or $t \rightarrow s$) where there is some other rule with s as its left or right-hand side, and where s is not in D.*

Lemma 2.4 *The processing of a bridging rule $s \rightarrow t$ always reduces the cardinality of D (since all occurrences of s are replaced by t, including in D). Also, processing any rule never increases the cardinality of D (obviously). Re-orienting the rules does not affect D, too.*

Lemma 2.5 *The processing of a linking rule always creates a bridging rule.*

Definition 2.6 *Given a ground system R, we define its* partition number *to be the cardinality of D.*

We use the partition number of R as $c(R)$, approximately. (It will be necessary to modify this measure later.) The processing of linking or bridging rules will always reduce the partition number, either immediately or on the next processing step, assuming that bridging rules are processed as soon as possible. We say that linking and bridging rules are *productive*, since they either reduce the partition number, or enable another rule to do so. We say that a rule is *unproductive* if it is not a bridging rule or a linking rule.

Now, it can happen that the processing of an unproductive rule may reduce the partition number by identifying terms of D. However, it is also possible that the processing of such unproductive rules does not change the partition number. This is so because a dominating set need only contain one of the sides of a rewrite rule. Thus the processing of a rule $r \rightarrow s$ may leave the cardinality of D unchanged, if $r \in D$ and $s \notin D$, for example.

Definition 2.7 *A rule $r \rightarrow s$ of R is* processable *if there are occurrences of r elsewhere in R. A rule is* potentially unproductive *if it's right-hand side is larger than or equal to the right-hand side of a processable unproductive rule.*

Definition 2.8 *We define* top(R) *to be the set of left and right-hand sides of rules in R. We say that a term u appears at the* top (level) of *a rule $r \rightarrow s$ if u is r or s.*

We observe that if a subterm u of R is not in $top(R)$, then $u \in D$.

Definition 2.9 *A* redex *of R is an occurrence of a subterm r of R which also appears on the left-hand side of some rule $r \rightarrow s$ of R; however, the occurrence of r in the rule $r \rightarrow s$ is not considered as a redex. If there is some other rule $r \rightarrow t$, then the occurrence of r in $r \rightarrow s$ is a redex.*

The idea is that an unprocessable potentially unproductive rule $r \rightarrow s$ can be made processable by processing an unproductive rule, since the orientation of the rule $r \rightarrow s$ may change. This rule can be rewritten to $r' \rightarrow s'$ with $r' < s'$; then we orient the rule to $s' \rightarrow r'$, and processing this rule may leave the partition number unchanged. On the other hand, if $r' > s'$ and the rule $r' \rightarrow s'$ can still be processed, then there must be a new redex, or an existing redex can be rewritten in a new way, which will eventually reduce the partition number. Therefore, the main problem as far as reducing the partition number, is dealing with rules that need re-orientation.

Lemma 2.10 *Processing a rule $r \rightarrow s$ can only change the orientation of a rule $u \rightarrow v$ in which $v > s$. It can only cause a rule $u \rightarrow v$ to be deleted if $v \geq s$.*

Proof. Suppose $r \rightarrow s$ is a rule in R. Let $u \rightarrow v$ be another rule, which is rewritten to $u' \rightarrow v'$ by the processing of $r \rightarrow s$. If $v \leq s$, then $v' \leq s$ also (since $v' \leq v$). If $u' \equiv u$, then $u' > v \geq v'$ (since $u > v$). If $u' < u$, then u' contains an occurrence of s, so $u' \geq s \geq v'$. In either case, $u' \geq v'$, and so the rule $u' \rightarrow v'$ will not need reorientation. By similar reasoning, we can only have $u' \equiv v'$ if $v \geq s$. $\square$

Definition 2.11 *Suppose that a rule $r \rightarrow s$ of R is processed, producing a system R'. Suppose u is a subterm of R, and during the processing of $r \rightarrow s$, u is rewritten to u' (or else u is not rewritten, and then u' is u). Suppose that u is not a redex of R, but u' is a redex of R'. Then we call u' a* new redex *in R'.*

Theorem 2.12 *Suppose that in the processing of a rule $r \rightarrow s$ of R, with dominating set D, creating the system R', with dominating set D', the two distinct subterms u and v of R are rewritten to the same term u' of R'. Suppose that $|D| = |D'|$. Then either a) there is a rule $u \rightarrow v$ or $v \rightarrow u$ in R, and $u' \in D'$, or b) there are rules $u \rightarrow w$ and $v \rightarrow w$ (possibly with different orientations) in R, and $w \in D$, or c) $u \in top(R)$ and $v \in top(R)$ and R' has a linking rule or a bridging rule, or d) exactly one of u and v is in $top(R)$, and R' has a bridging rule.*

Proof. Note that if $u \in D$ and $v \in D$ then D' has fewer elements than D. By the hypothesis $|D| = |D'|$, we know that there are no two elements of D that map to the same element of D'. For case a), if there is a rule $u \rightarrow v$ or $v \rightarrow u$ in R, then either u or v is in D, so $u' \in D'$. For case b), if there are rules $u \rightarrow w$ and $v \rightarrow w$ (possibly with different orientations) and $w \in D$, then both of these rules will rewrite to $u' \rightarrow w'$, and one (copy) of these rules will be deleted in R'. For case c), if both u and v are in $top(R)$, then they must appear at the top of two different rules of R, since we have already considered the case where there is a rule $u \rightarrow v$ or $v \rightarrow u$. Therefore, these two rules of R have between them at least two elements of D at the top level (since in case b) we have eliminated the case where there is a common element). These two rules may have a single successor in R', which is then a bridging rule. Otherwise, we obtain two rules of R' which have among them two distinct elements of D' (since the hypothesis $|D| = |D'|$ excludes the case in which two elements of D map to the same element). However, these two rules of R share a top-level element (left or right-hand side). So there must be a linking rule or a bridging rule in R'. Now, a) covers the case where u and v appear at the top of the same rule. Case b) covers the case where they appear at the top of different rules with a common top element. Case c) covers the case where they appear at the top of different rules with no common elements. If neither u nor v appears at the top of a rule of R, then they are both in D, which implies $|D'| < |D|$, excluded by the hypothesis $|D| = |D'|$. The only other case is where one of u and v appears at the top level of a rule and the other does not. Suppose without loss of generality that u is not in $top(R)$ but there is a rule $v \to w$ (or $w \to v$) in R. We already excluded the case where u and v are both in D, so we know v is not in D, so w is in D. Therefore in R' we have a rule $u' \to w'$ or $w' \to u'$ in which both u' and w' are in D'; this is a bridging rule, as claimed. □

We note that in all cases except case a) and b) of this theorem, if the cardinality of D' is not less than the cardinality of D, then R' has a linking or a bridging rule, which permits the cardinality of D' to be reduced in one or two more processing steps.

Corollary 2.13 *Suppose $u \to v$ is a rule of R that is not processable. Suppose that some other rule of R is processed, yielding the system R', and the rule $u \to v$ is processed to the rule $u' \to v'$ (that is, u rewrites to u' and v rewrites to v'). Suppose that $u' > v'$, and the rule $u' \to v'$ is processable in R'. Then the partition number of R' is less than that of R, or else R' has a linking rule or a bridging rule, or else there is some other rule $t \to v$ (or $v \to t$) of R that is processable in R and rewrites to $u' \to v'$ under processing.*

Proof. Since $u \to v$ was not processable, there were no other occurrences of u in R. Since $u' \to v'$ is processable, there is some other occurrence of u' in R'. Therefore there is some other term t in R that rewrites to some other occurrence of u' under processing. By the preceding theorem, either there was a rule $t \to u$ or $u \to t$ in R, or else rules of the form $t \to w$ and $u \to w$, et cetera, or R' has a linking or a bridging rule, or else D' has fewer elements than D. However, there were no other occurrences of u in R, so there could be no rule $t \to u$ or $u \to t$ in R. The only possibility is that there were rules $t \to w$ and $u \to w$ in R (possibly with different orientations), for some term w. But since there is no other occurrence of u in R, we must have that the rules $u \to w$ and $u \to v$ are identical, that is, $w \equiv v$. Thus there were rules $t \to v$ (or $v \to t$) and $u \to v$ in R. For $u' \to v'$ to be processable in R', there must be some other occurrence of t in R, not at the top level of a rule, that rewrites to another occurrence of u' in R'. This implies that the rule $t \to v$ was processable in R. Or, if this rule was oriented $v \to t$, then it was processable on the rule $u \to v$. Note that the rule $t \to v$ (or $v \to t$) rewrites to $u' \to v'$ under processing. This completes the proof of the corollary. □

Corollary 2.14 *If the processing of an unproductive rule $r \to s$ with a minimal right-hand side (s) enables the processing (or reprocessing) of a rule $u' \to v'$ with $v' \leq s$, then the partition number was decreased, or else a bridging or a linking rule was created, or else the number of potentially unproductive rules in R' is smaller than in R.*

Proof. We are assuming that the rule $u' \to v'$ comes from a rule $u \to v$ of R that was not processable before, or was just processed (i.e., it is the rule $r \to s$). Suppose $v' < s$. Then the rule $u \to v$ was not just processed. In this case, the rule $u' \to v'$ does not require re-orientation, by lemma 2.10. By corollary 2.13, the partition number of R' is less than that of R, or else R' has a linking rule or a bridging rule, or else there is some other rule $t \to v$ (or $v \to t$) of R that is processable in R and rewrites to $u' \to v'$ under processing. Suppose this rule was oriented $t \to v$; since it was processable in R, $v \geq s$ (by the hypotheses of the corollary). Therefore $v' \geq s$, and we are assuming $v' < s$. This is a contradiction. Suppose now that this rule was oriented $v \to t$. Since $u \to v$ was a rule in R, $v \to t$ was processable. This again implies that $t \geq s$, so $v > s$, hence $v' \geq s$, contradiction.

Suppose now that $v' \equiv s$ and that the rule $u \to v$ was not just processed. Then, by lemma 2.10, $u' \geq v'$ also. Reasoning as above, either the partition number of R' is less than that of R, or else R' has a linking rule or a bridging rule, or else there is some other rule $t \to v$ (or $v \to t$) of R that is processable in R and rewrites to $u' \to v'$ under processing. If this rule is oriented $t \to v$, then since $v \geq s$, this rule is potentially unproductive, and the number of potentially unproductive rules of R' is reduced. If this rule is oriented $v \to t$, the same is true, by the hypotheses to the corollary (since we process a rule with a minimal right-hand side).

If this rule $u' \to v'$ was just processed, then $u \equiv u' \equiv r$ and $v \equiv v' \equiv s$, since a rule cannot rewrite itself. If this rule is still processable, then there must be a term t other than u in R that rewrites to u under processing, since all other occurrences of u in R were eliminated. Furthermore, the rule $u' \to v'$ must still be processable on an occurrence of u obtained from t. If there are no occurrences of the rules $t \to u$ or $u \to t$ in R, or for some term w in D the rules $t \to w$ and $u \to w$ (possibly with other orientations), then the result follows by theorem 2.12.

Otherwise, (that is, $u' \to v'$ was just processed, and at least one of the rules $t \to u$ etc. exist in R), we note that t contains one or more occurrences of u, since t rewrites under the rule $u \to v$. Therefore we write t as $t[u]$. Now, t rewrites to u, hence u is $t[v]$, and t is $t[t[v]]$. We consider all of the rules $t \to u$, $u \to t$, $t \to w$ ($w \to t$), or $u \to w$ ($w \to u$) that might have been in R. Note that the first two of these rewrite to $u \to v$, and thus duplicate another rule of R', and will be deleted. For example, consider the rule $t \to u$; this is $t[t[v]] \to t[v]$ and rewrites to $t[v] \to v$, that is, to $u \to v$. Thus this occurrence of $t[u]$ does not satisfy our hypothesis about rewriting to an occurrence of $t[v]$ that enables a reprocessing of the rule $u' \to v'$.

Consider the rules $t \to w$ and $u \to w$ (possibly with other orientations), with w in D. These rewrite to $u \to w'$ and $s \to w'$ (possibly with other orientations) if $w \not\equiv v$, with w' in D'. We must have $s \in D$, since the rule $r \to s$ was just processed. Therefore the rule $w' \to s$ is a bridging rule. Or it can happen that $w \equiv v(\equiv s)$, in which case $w' \equiv s$. In this case, the rule $u \to w'$ becomes $u \to s$, that is, $u \to v$, and will be deleted, since there is already a copy of this rule in R'. Therefore, the number of potentially unproductive rules decreases. This completes the proof of corollary 2.14.

$\square$

Lemma 2.15 *If the processing of all of the k processable unproductive rules $r \to s$ with minimal right-hand side s does not reduce the partition number and does not create bridging or linking rules, then the number of potentially unproductive rules has been reduced by k. Here k includes the rules that rewrite during this processing so* *that their right-hand side is s, as well as rules that have s as a right-hand side at the beginning.*

Proof. By the proof of the preceding corollary, this means that no rules with right-hand sides smaller than or equal to s, can be processed. Also, after all such rules $r \to s$ have been processed, then the smallest processable unproductive rule has a right-hand side larger than s. Therefore, all of the k rules $u \to v$ with $v \equiv s$ now cease to be potentially unproductive. This processing is done by processing all the rules with s on the right-hand side, one by one, until all (at most k) of these processable rules have been processed. If during this processing, some formerly unprocessable rule or previously processed rule becomes again processable, then by corollary 2.14, the partition number is decreased, or else a linking or a bridging rule is created, or the number of potentially unproductive rules is reduced. Otherwise, after at most k processing steps, all rules with right-hand side s can be processed, and the number of potentially unproductive rules is reduced by k. □

We note that the processing of all such rules with minimal right-hand side, can be done in polynomial time. Each rule can be processed in polynomial time, and there are a linear number of such rules. It is possible for a rule $r \to s$ to need processing more than once, if s is a proper subterm of r, but this can only happen a finite number of times, since each such step reduces the size of R. Furthermore, in this case, the processing of this rule is creating new occurrences of r, that is, new redexes, so the partition number decreases, or else linking or bridging rules are created, or the number of potentially unproductive rules decreases. An example is the rule $f(c) \to c$ applied to a term $f(f(c))$, which we assume is not a redex. After one application, we have $f(f(c))$ rewritten to $f(c)$, which is a redex, so a new redex has been created, and the partition number is reduced, or else new linking or bridging rules are created, or the number of potentially unproductive rules is reduced.

Our method, then, is as follows.

2.1 Method A (Polynomial Ground Completion)

1. Whenever a rule is rewritten, it is oriented so that its left-hand side is larger than its right-hand side. This orientation is not counted as a separate step. Whenever a rule is oriented, it need not be processed.

2. We always prefer to process a bridging rule (since it reduces the cardinality of D), then, if there are no bridging rules, a linking rule (since it creates a bridging rule), then any processable rule with a minimal right-hand side.

2.2 The Time Required by Method A

Lemma 2.16 *The partition number will never increase in this scheme.*

It is clear that processing of bridging rules and linking rules will reduce the partition number. When we process a rule with the smallest right-hand side, we may leave the partition number unchanged. However, whenever we process all rules with the smallest right-hand side, we reduce the number of potentially unproductive rules, as we showed in lemma 2.15. Thus the unproductive rule applications are limited in this way; after a linear number of such processing steps, either the method terminates, or we have created a new bridging rule or linking rule.

Definition 2.17 *We let $c(R)$ be $(p-1)(n+2)+u$, where p is the partition number of R, n is the number of rules of R, and u is the number of potentially unproductive rules of R.*

Theorem 2.18 *This measure $c(R)$ is quadratic in the size of the system, non-negative, and decreases by at least k whenever the (k) processable unproductive rules with the smallest right-hand side are processed assuming no linking or bridging rules are created, decreases by at least two when a bridging rule is processed, and decreases by at least two when a linking rule and then a bridging rule are processed.*

Proof. The measure $c(R)$ is quadratic in the size of R because p and n are linear in the size of R. Also, $c(R) \geq 0$ because $p \geq 1$. When all (k) processable unproductive rules with the smallest right-hand side are processed, the number of potentially unproductive rules decreases by k, or else the partition number is reduced by at least one, or else a linking or a bridging rule is created, by lemma 2.15. Reducing the number u of potentially unproductive rules reduces $c(R)$ immediately. Suppose R' has a smaller partition number p'; then $c(R') = (p'-1)(n'+2)+u'$ and $c(R) = (p-1)(n+2)+u$. Thus $c(R)-c(R') \geq (n+2)+(u-u')$. Since $u-u' \geq -n$, $c(R)-c(R') \geq 2$. Processing a bridging rule reduces the partition number, and thus reduces $c(R)$ by at least 2, as just shown. Similary, processing a linking rule and then a bridging rule reduces $c(R)$ by at least two, since the partition number is reduced. □

Corollary 2.19 *Method A has a quadratic worst-case bound on the number of rule applications required to complete a ground system, and a linear bound on the number of applications of linking and bridging rules.*

Proof. Each processing of the (k) processable unproductive rules with smallest right-hand side reduces $c(R)$ by at least k, unless a linking or a bridging rule is created; processing a bridging rule reduces $c(R)$ by at least two; and the pair of processing steps resulting when a linking rule is created and then a bridging rule, reduces $c(R)$ by at least two. Processing the k rules with smallest right-hand side and then processing a bridging rule, possibly processing a linking rule in between, reduces $c(R)$ by at least $k+2$. This follows because we have $c(R)-c(R') = (p-1)(n+2)+u-((p'-1)(n'+2)+u')$ and $p' < p$ and $n' \leq n$, so $c(R)-c(R') \geq (n+2)+(u-u')$. However, $u' \leq n' \leq n$ and $u \geq k$, so $u-u' \geq k-n$, and $c(R)-c(R') \geq 2+k$. Thus each processing step reduces $c(R)$ by an average of at least one. Since $c(R)$ is non-negative, the total number of processing steps is bounded by $c(R)$, which is quadratic in the size of the initially given system R. We can also see this in another way. Processing all rules with a minimal right-hand side requires only one processing step per rule, so after a number of processing steps bounded by the number of rules in R, either the partition number is decreased, or a new linking or bridging rule is created, or else the method terminates. Let's define a *phase* to be a step in which all the processable unproductive rules with minimal right-hand sides are processed; a phase ends when the partition number decreases or a new linking or bridging rule is created. Each phase has at most n steps, and the number of phases is at most $p - 1$, since $p \geq 1$ always. Also, in each phase, there may be some number of additional processing steps, to deal with the linking or bridging rules that may have been created. The total number of steps for unproductive rules is at most $(p-1) * n$. We note that p is bounded by the number of distinct subterms in R, irrespective of how many times they occur. The total number of steps processing bridging rules is at most $p - 1$, since each such step reduces the partition number by one. Also, the total number of steps processing linking rules is at most $p - 1$, since each such step creates a bridging rule. Therefore the total number of steps is bounded by $(p-1)*n+(p-1)+(p-1)$, or, $(p-1)*(n+2)$, and is quadratic in the size of R. We note that $c(R) \geq (p-1)*(n+2)$. Thus after at most $c(R)$ processing steps, R will be completed. We also note that the number of steps involving the processing of bridging and linking rules is at most linear in the size of R, so that if there are few unproductive steps, then the bound is linear instead of quadratic. We don't know whether the quadratic bound can be achieved, that is, whether an example exists with quadratic behavior, or if in fact a linear bound holds in general. $\square$

This quadratic bound is not as efficient as congruence closure, but not too bad, especially if the number of applications of unproductive rules is small and the data structures are efficient. The moral seems to be that top-level rewrites are good, since linking rules are often top-level rewrites, and after that, applications with small right-hand sides are good. Bridging rules are good, but they are hard to detect unless one explicitly keeps track of D. Also, a rule should be applied to all possible redexes at the same time. In addition, it is important to use an efficient directed acyclic graph representation, in order to be able to process terms efficiently.

We now present another method which is simpler and also has polynomial behavior.

2.3 Method B (Polynomial Ground Completion)

1. Whenever a rule is rewritten, it is oriented so that its left-hand side is larger than its right-hand side. This orientation is not counted as a separate step. Whenever a rule is oriented, it need not be processed.

2. We always process a rule (any processable rule) with a minimal right-hand side.

2.4 The Time Required by Method B

To show that this is quadratic, we introduce some lemmas and definitions.

Definition 2.20 *$St(R)$ is the set of subterms appearing in rules of R.*

Lemma 2.21 *Suppose processing a rule $r \rightarrow s$ of R produces R'. Then $|St(R')| \leq |St(R)|$.*

Proof. Let us express subterms of R as $t[r]$, indicating all the occurrences of r, if any. Then $St(R) = \{r\} \cup \{t[r] \in St(R) : t[r] \not\equiv r\}$ and $St(R') = \{r\} \cup \{t[s] : t[r] \in St(R), t[r] \not\equiv r\}$. The term r remains in R', since the rule $r \rightarrow s$ is still present in R'. We note that $St(R')$ is obtained by a mapping of $St(R)$, therefore, $|St(R')| \leq |St(R)|$. $\square$

Lemma 2.22 *Suppose u and v are two distinct subterms of R and both rewrite to u' when the rule $r \rightarrow s$ is processed. Suppose at least one of these terms is rewritten but not at the top level, that is, at least one of u and v has r as a proper subterm. Then $|St(R')| < |St(R)|$.*

Proof. Recall that $St(R) = \{r\} \cup \{t[r] \in St(R) : t[r] \not\equiv r\}$ and $St(R') = \{r\} \cup \{t[s] : t[r] \in St(R), t[r] \not\equiv r\}$. Suppose u has r as a proper subterm; then u is in $\{t[r] \in St(R) : t[r] \not\equiv r\}$. If v also has r as a proper subterm, or is distinct from r, then, since u and v rewrite to the same term, $|\{t[s] : t[r] \in St(R), t[r] \not\equiv r\}| < |\{t[r] \in St(R) : t[r] \not\equiv r\}|$. Thus $|St(R')| < |St(R)|$. If v is r, then a similar argument applies, since r will be an element of $\{t[s] : t[r] \in St(R), t[r] \not\equiv r\}$. $\square$

Lemma 2.23 *Suppose the rule $r \rightarrow s$ of R is a processable rule with a minimal right-hand side s. Suppose that some other rule $u \rightarrow v$ of R with $v < s$, rewrites to $u' \rightarrow v'$, which is processable in R'. Then $|St(R')| < |St(R)|$.*

Proof. Note that $v' \equiv v$. Since $u \rightarrow v$ is not processable in R, there are no other occurrences of u in R. But there are other occurrences of u' in R'. Therefore, there must be some other term w in R that rewrites to u' under processing. Now, we cannot have $u \equiv r$, for this would imply that the rule $u \rightarrow v$ (i.e., $r \rightarrow v$) was processable in R, on the rule $r \rightarrow s$. If $w \equiv r$, then $u' \equiv s$, implying that $u \equiv r$ or $u \equiv s$. As already shown we cannot have $u \equiv r$. Moreover, u cannot be s, since this would imply that the rule $u \rightarrow v$ was processable in R on the rule $r \rightarrow s$. Therefore neither rewrite is at the top level, so by lemma 2.22, $|St(R')| < |St(R)|$. $\square$

Lemma 2.24 *Suppose the rule $r \to s$ of R is a processable rule with a minimal right-hand side s. Suppose that some other rule $u \to v$ of R with $v \geq s$, is not processable in R, and rewrites to $u' \to v'$, which is processable in R'. Suppose $u' > v'$ and $v' \equiv s$. Then $|St(R')| < |St(R)|$.*

Proof. Since $u \to v$ is not processable in R, there is no other occurrence of u in R. However, there is another occurrence of u' in R'. Therefore, there is some other term w in R that rewrites to u' in R'. Now, w cannot be r, since that would imply that u' is s. But v' is s, which means that the rule $u' \to v'$ would be $s' \to s'$, which would be deleted, and not processable. Also, u cannot be r, since this would imply that u' is s, which we have just excluded. Therefore neither rewrite of u nor w is at the top level, and both rewrite to the same term u'. By lemma 2.22, $|St(R')| < |St(R)|$. □

Theorem 2.25 *Method B completes a ground system R in a quadratic number of processing steps.*

Proof. Consider the measure $c'(R) = u + |St(R)| * (n + 1)$. Recall that u is the number of potentially unproductive rules, and n is the number of rules altogether. Now, suppose that we process a rule $r \to s$ of R, obtaining R'. We claim that each processing step reduces $c'(R)$ by an average of at least one. Also, we note that $c'(R)$ is quadratic in the size of R.

If $|St(R')| < |St(R)|$, then $c'(R') < c'(R)$, since $u \leq n$. If $|St(R')| = |St(R)|$ (which is the only other possibility, by lemma 2.21), then the processing of $r \to s$ cannot make any rule $u \to v$ with a smaller right-hand side v, processable, by lemma 2.23. Also, it cannot make any formerly unprocessable rule $u \to v$ processable, with $v' \equiv s$, by lemma 2.24. It can be that the rule $r \to s$ is still processable again after it is processed once; this can happen if s is a proper subterm of r. We can express r as $r[s]$; then there must be some term $r[r[s]]$ that rewrites to $r[s]$. By lemma 2.22, since r also occurs on the left-hand side of the rule $r \to s$, $|St(R')| < |St(R)|$ if this happens.

Now, assume that all the processable rules with s on the right-hand side are processed, one by one, and none of these can be processed more than once. Here we also include rules that are rewritten so that their right-hand side is s, and process these, too, where possible. If $|St(R)|$ does not decrease, this means that none of these rules are processable more than once. Therefore, when all of these (k) processable rules with minimal right-hand side s are processed, if $|St(R)|$ does not decrease, then u decreases by at least k. Thus, on the average, $c'(R)$ decreases by at least one with every processing step. Therefore, after at most $c'(R)$ steps, the system R will be completed. Also, $c'(R)$ is quadratic in the size of R, since u, $|St(R)|$, and n are linear in the size of R. □

We note that this method and proof are substantially simpler than Method A and its proof. But we still think that Method A is interesting. Since Method A always processes bridging and linking rules when they exist, it may tend to terminate faster. We could also modify Method B to prefer to process rules that reduce $|St(R)|$, but it's not clear that such rules are easy to detect rapidly.

3 Non-Ground Systems

In the non-ground case, we consider *critical pair proofs*; these are sequences of equations and inequations in which each equation or inequation is either given or is a critical pair obtained from earlier equations in the proof. For this, it is also necessary to generalize the concept of a critical pair operation to inequations, as we will do below. Also, we allow these proofs to contain a fairly powerful rewriting operation on equations, as in the ground case. We want to say something about the lengths of these critical pair proofs, that is, the number of critical pair operations needed to derive contradictions, which in this context are equations of the form $u \neq u$ for some term u. The methods already described for completing ground systems have some implications for non-ground systems, too. That is, we can say something about the lengths of critical pair proofs from sets of equations and inequations possibly containing variables. In particular, we obtain a polynomial bound on proof length in terms of the number g of distinct ground instances of equations and inequations (as in Herbrand's theorem) that are actually used in a proof. This result is somewhat surprising, because it only depends on the number g of ground instances needed for the proof, and not on their size, which can be exponential in g. Furthermore, these ground instances are not given initially, but must be constructed incrementally in some manner during the critical pair proof. For non-ground systems, as for ground systems, it is necessary to use a rewriting operation that applies to all occurrences of a subterm at the same time in order to obtain a polynomial bound on proof length. As in the ground case, one must use a directed acyclic graph representation in order to implement this kind of rewriting efficiently. Also, we cannot allow arbitrary rewriting (reduction) operations, since these can complicate the proof. Instead, reduction must be carefully controlled, as in the polynomial ground completion method.

In the non-ground case, we consider the problem of theorem proving, rather than completion per se. This permits our results to apply to arbitrary sets of equations, including those that cannot be completed. For discussions of mechanical theorem proving, see [Lov78, CL73, WOLB84]. We first make some general comments about refutational theorem proving. Theorem proving is often expressed in terms of unsatisfiability of sets of formulae. We know by Herbrand's theorem that if A is an unsatisfiable set of clauses, then there is a finite unsatisfiable set T of ground instances of A. For our purposes, the only clauses we will consider are equations, inequations, and equality axioms. If A is $\{C_1, C_2, \cdots, C_n\}$, then T may be expressed as $\{C_{i_1}\Theta_1, C_{i_2}\Theta_2, \cdots, C_{i_k}\Theta_k\}$ where all $C_{i_j}\Theta_j$ are ground clauses. We say that two clauses are *variants* if they are instances of each other, that is, one is obtained from the other by a renaming of variables. We can then let $\{C'_1, C'_2, \cdots, C'_k\}$ be clauses where each C'_j is a variant of C_{i_j}, and where no variable appears in more than one clause C'_j; such a set $\{C'_1, C'_2, \cdots, C'_k\}$ is called an *amplification* of A. It follows that there is a Θ such that $\{C'_1, C'_2, \cdots, C'_k\}\Theta = T$; we can choose Θ so that $C'_j\Theta \equiv C_{i_j}\Theta_j$. In this way, we can show in general that if A is unsatisfiable, then there is an amplification A' of A and a substitution Θ such that $A'\Theta$ is ground and unsatisfiable. Such amplifications are essential to our construction. We note that there is no recursive way to construct this amplification from A, because of the undecidability of first-order logic. However, this approach still gives a convenient basis for complexity arguments and for the comparison of different methods.

In this paper, we are interested in unsatisfiability relative to equality. For a general set A of formulas, we say that A is unsatisfiable relative to the equality axioms if $A \cup Eq$ is unsatisfiable, where Eq are the usual equality axioms (reflexivity, symmetry, transitivity, and functional replacement). We note that the axiom $s = t \supset s\Theta = t\Theta$ follows implicitly from the rules of first-order logic. By Herbrand's theorem, if A is unsatisfiable relative to the equality axioms, then there is a finite set of ground instances of A that is also unsatisfiable relative to the equality axioms. The problem we consider is to determine whether $R \models s = t$, or equivalently, whether $R \cup \{s \neq t\}$ is unsatisfiable relative to the equality axioms. If $R \cup \{s \neq t\}$ is unsatisfiable relative to the equality axioms, then there is a set T of ground instances of formulae in $R \cup \{s \neq t\}$ such that T is unsatisfiable relative to the equality axioms. Then g (mentioned above) may be taken as the cardinality of T. Note that it is possible for more than one ground instance of a rule to be used. That is, the cardinality g of T can be larger than the cardinality of $R \cup \{s \neq t\}$. However, since the equality axioms are Horn clauses (that is, they contain at most one positive literal), it turns out that we need only one instance $s' \neq t'$ of the inequation $s \neq t$. This also follows from Birkhoff's theorem, mentioned below. We define an *amplification* of R to be a finite set R_1 of copies of the elements of R such that the variables in different copies have been renamed, so that no two equations in R_1 share a variable. From the existence of T and above comments, it follows that there exists an amplification R_1 of R and a ground substitution Θ such that $(R_1 \cup \{s \neq t\})\Theta$ is unsatisfiable; this exists because there is a finite set R_1 and a ground substitution Θ such that $(R_1 \cup \{s \neq t\})\Theta = T$. Thus we have something of the flavor of rigid E-unification [GNRS92, GNPS90]. In general we use the term *rigid* to refer to the fact that variables cannot be renamed at will, but are global to R_1. We use the term *non-rigid* to indicate that free variables are local to each equation or clause, and may be renamed within each equation or clause. We process this amplification R_1 of R in a rigid manner to obtain a proof whose size is polynomial in g. Then we convert the proofs obtained into non-rigid proofs from $R \cup \{s \neq t\}$. Of course, in practice we do not know T, but its existence enables us to obtain a bound on proof size. In the following discussion we let S be $R_1 \cup \{s \neq t\}$. We assume here that $>$ is an arbitrary but fixed total ordering on ground terms, as in the discussion of ground systems; thus $>$ must be well-founded and monotonic.

Definition 3.1 *If Θ is a substitution, then a* Θ-term *is a term $x\Theta$ for some x such that $x\Theta \not\equiv x$.*

Definition 3.2 *A substitution Θ is* ground *if for each variable x, either $x\Theta \equiv x$ or $x\Theta$ is a ground term, that is, a term containing no variables.*

Definition 3.3 *Suppose E is a set of ground equations. Let R be the term-rewriting system $\{r \to s : r > s, r = s \in E \text{ or } s = r \in E\}$. We say a term t is E-reducible if t is R-reducible. A ground substitution Θ is* E-reducible *if there is a variable x such that* *the term $x\Theta$ is E-reducible, that is, $x\Theta$ has a subterm r and there is some equation $r = s$ or $s = r$ in E with $r > s$. Such an equation can be used as a reduction to replace r by s. We say Θ is E-*irreducible *if it is not E-reducible. If S is a set of ground equations and inequations, we say that Θ is S-reducible iff it is E-reducible, where E is the set of equations in S. We define the rewrite relation $\rightarrow_E$ and $\rightarrow_S$ to be identical to $\rightarrow_R$.*

Definition 3.4 *Recall that $<$ is a total termination ordering on ground terms. Recall that $r \leq s$ if $r < s$ or $r \equiv s$. We order ground substitutions Θ, Θ' by $\Theta \leq \Theta'$ if for all variables x, $x\Theta \leq x\Theta'$. We say $\Theta < \Theta'$ if $\Theta \leq \Theta'$ and $\Theta \not\equiv \Theta'$. We note that this is a well-founded ordering on substitutions.*

Theorem 3.5 *Suppose S is a set of equations and inequations and for some ground substitution Θ, $S\Theta$ is ground and unsatisfiable relative to the equality axioms. Then either S contains an equation of the form $y = t$ or $t = y$ where t is a term not containing y, or there exists a ground substitution Θ' such that Θ' is $S\Theta'$-irreducible and $S\Theta'$ is ground and unsatisfiable relative to the equality axioms.*

Proof. By induction on Θ in the well-founded ordering on substitutions given earlier. Suppose that S contains no equation of the form $y = t$ or $t = y$ where t is a term not containing y. Suppose $S\Theta$ is ground and unsatisfiable relative to equality and Θ is not $S\Theta$-irreducible. Let Θ_1 be obtained by reducing Θ one reduction step using $\rightarrow_{S\Theta}$. That is, for some variable x, $x\Theta \rightarrow_{S\Theta} x\Theta_1$. Suppose the equation $r\Theta = s\Theta$ was used to perform this reduction, with $r\Theta > s\Theta$ and the equation $r = s$ or $s = r$ in S. Now, r cannot be a variable, since $r\Theta > s\Theta$ then implies that this variable does not occur in s, and we excluded equations of the form $y = t$ or $t = y$, where y does not appear in t. We claim that $r\Theta_1 \equiv r\Theta$ and $s\Theta_1 \equiv s\Theta$. The first part follows because $r\Theta_1$ is only different from $r\Theta$ if the rule $r\Theta \rightarrow s\Theta$ applied to a proper subterm of $r\Theta$, and $r\Theta$ cannot be a proper subterm of itself. The second part follows because the ordering $>$ is well-founded; if $s\Theta$ had a subterm $r\Theta$, then $\rightarrow_{S\Theta}$ would not terminate. Therefore the equation $r\Theta = s\Theta$ is still present in the system $S\Theta_1$. This implies that $S\Theta_1 \models S\Theta$. Since $S\Theta$ is unsatisfiable relative to the equality axioms, so is $S\Theta_1$. Since $\Theta_1 < \Theta$, we can assume by induction that there is a Θ' as in the theorem. This completes the proof. $\square$

Example 3.6 Suppose $S = \{f(x) = g(x), f(y) = g(y)\}$ and $\Theta = \{x \leftarrow g(b), y \leftarrow b\}$ and the ordering $>$ is the length-lexicographical ordering induced by $g > f > b$. Then Θ is $S\Theta$-reducible, since $S\Theta = \{f(g(b)) = g(g(b)), f(b) = g(b)\}$, so the term $g(b)$ in Θ can be reduced using the equation $f(b) = g(b)$, noting that $g(b) > f(b)$. Then we can let Θ_1 be Θ with this occurrence of $g(b)$ replaced by $f(b)$; thus, $\Theta_1 = \{x \leftarrow f(b), y \leftarrow b\}$. Note that $\Theta_1 < \Theta$, and in this case Θ_1 is $S\Theta_1$-irreducible. Also, $S\Theta_1$ is logically equivalent to $S\Theta$, since $S\Theta_1$ still contains the equation $f(b) = g(b)$ which, by applying it in the forward direction, can be used to derive $S\Theta$ from $S\Theta_1$.

Definition 3.7 *We say that a set S of equations and inequations is* rigid reducible *relative to a ground substitution Θ if there is an equation $r = s$ or $s = r$ in S with $r\Theta > s\Theta$ and there is another occurrence of the subterm r in S. In the following we will also say that the term r is rigid reducible in this case. A* rigid reduction *(or* rewrite*) is then a replacement of some other occurrence of r by s. Note that this other occurrence of r must have exactly the same variables as r does.*

Lemma 3.8 *Suppose S is rigid reducible relative to Θ. Let $r = s$ be an equation in S with $r\Theta > s\Theta$, and let S' be S with all other occurrences of r replaced by s. That is, S' is obtained by a sequence of rigid reductions on S. Then S and S' are logically equivalent, and also $S\Theta$ and $S'\Theta$ are logically equivalent.*

Proof. The equation $r = s$ is still present in S', since the top-level occurrence of r in it was not replaced, and there can be no other occurrences of r in it because $r\Theta > s\Theta$. Therefore, using this equation in the reverse direction, S can be derived again from S' by reversing all rewrites performed. Thus S' logically implies S. The fact that S' was derived from S by rewriting shows that S logically implies S'. Thus S and S' are logically equivalent. This implies that $S\Theta$ and $S'\Theta$ are logically equivalent. □

Definition 3.9 *A* critical substitution *for S and a ground substitution Θ is a most general unifier of the left or right-hand side of some equation $r = s$ of S, with some other subterm in S. We assume that the side used is the larger of $r\Theta$ and $s\Theta$ in the termination ordering $>$.*

Definition 3.10 *The* subterm size *of a set S of clauses is the cardinality of the set of subterms of S, that is, $|St(S)|$. Thus we only count the subterms, and not how often they occur.*

Lemma 3.11 *Suppose α is a critical substitution for S and Θ. Then $|St(S\alpha)| \leq |St(S)|$.*

Proof. By examining a unification algorithm, we can express α as a composition of substitutions of the form $x \leftarrow t$, where the variable x does not appear in the term t, and x and t are subterms of $S\beta$, and β is the composition of the previous substitutions applied to S. It suffices then to show that $|St(S\beta\{x \leftarrow t\})| \leq |St(S\beta)|$ for $t \in St(S\beta)$. We note that $St(S\beta\{x \leftarrow t\}) = \{u\{x \leftarrow t\} : u \in St(S\beta)\} \cup St(t)$. However, since t appears in $S\beta$, and x does not appear in t, $t\{x \leftarrow t\} \equiv t$ and $St(t) \subseteq St(S\beta\{x \leftarrow t\})$. Thus $St(S\beta\{x \leftarrow t\}) = \{u\{x \leftarrow t\} : u \in St(S\beta)\}$, and the lemma follows. □

Definition 3.12 *Suppose $r_1 = s_1$ (or $s_1 = r_1$) and $r_2 = s_2$ (or $s_2 = r_2$) are two equations in S. Suppose r_1 has a subterm t that unifies with r_2; thus, $r_1 \equiv r_1[t]$. Let α be a most general unifier of t and r_2. Then we call the pair $(r_1[s_2]\alpha, s_1\alpha)$ of terms a* critical pair*; we view the equation $r_1[s_2]\alpha = s_1\alpha$ as being derived from the two equations $r_1 = s_1$ and $r_2 = s_2$ by one critical pair operation. If $r_1 \neq s_1$ is an inequation with such a subterm t, then we consider that the inequation $r_1[s_2]\alpha \neq s_1\alpha$ can be derived from the inequation $r_1 \neq s_1$ and the equation $r_2 = s_2$ by one critical pair operation, also.*

We note that the critical pair $(r_1[s_2]\alpha, s_1\alpha)$ can be derived by applying the substitution α to both equations, and then performing a rigid reduction. So we will often just apply substitutions in our proofs, and perform the rigid reductions separately.

Theorem 3.13 *Suppose that S is a set of equations and one inequation. Suppose Θ is an $S\Theta$-irreducible ground substitution such that $S\Theta$ is ground and unsatisfiable (relative to equality). Then either*

1. *There exist unifiable terms u and v such that S contains an inequation $u \neq v$, or*
2. *S is rigid reducible relative to Θ, or*
3. *S has a critical substitution α (for Θ) which binds at least one variable of S to a term not containing that variable, and such that $S\Theta$ is an instance of $S\alpha$.*

Proof. Let R be the set of rewrite rules $\{r\Theta \rightarrow s\Theta : r\Theta > s\Theta, r = s \in S \textit{ or } s = r \in S\}$. If R is not confluent, it must have a critical pair. The overlap cannot be within a Θ-term, since these terms are $S\Theta$-irreducible (and hence R-irreducible). Suppose the overlap is between rules $r_1 \rightarrow s_1$ and $r_2 \rightarrow s_2$. Suppose r_2 is a subterm of r_1. (Recall that R is a ground system.) These rules are instances of equations $r'_1 = s'_1$ (or $s'_1 = r'_1$) and $r'_2 = s'_2$ (or $s'_2 = r'_2$) of S, respectively; that is, $(r'_1 = s'_1)\Theta \equiv (r_1 = s_1)$, and $(r'_2 = s'_2)\Theta \equiv (r_2 = s_2)$. The position γ of r_2 in r_1 must be a non-variable position of r'_1, since Θ-terms are $S\Theta$-irreducible. Let α be the unifier of r'_2 and the γ subterm of r'_1. This is a critical substitution for S and Θ. If this unifier α is trivial, then r'_2 and hence S is rigid reducible relative to Θ (case 2 of the theorem). Otherwise, this unifier α binds a variable of S to a term not containing this variable, and $S\Theta$ is an instance of $S\alpha$ (case 3 of the theorem).

Now, suppose R is confluent. Let $s_1 \neq s_2$ be the inequation in S. Then $s_1\Theta$ and $s_2\Theta$ have a common R-normal form. If $s_1\Theta \equiv s_2\Theta$, then s_1 and s_2 are (rigid) unifiable, and case 1 of the theorem is satisfied. Otherwise, at least one of $s_1\Theta$ and $s_2\Theta$ is R-reducible. If either s_i is rigid reducible relative to Θ, then case 2 of the theorem is satisfied. Otherwise, we can reduce s_1 and s_2 to a common term by "narrowing," since their instances $s_i\Theta$ reduce to a common term under R. Such a narrowing step corresponds to an overlap between $s_1 \neq s_2$ and R, and this produces a unifying substitution α as above that satisfies the conditions of case 3 of the theorem. This completes the proof. Note that we permit critical pairs involving an equation and an inequation. □

Corollary 3.14 *Suppose that S is a set of equations and an inequation and Θ is a ground substitution such that $S\Theta$ is ground and unsatisfiable relative to the equality* *axioms. Suppose we consider proofs consisting of sequences S_i of sets, where $S_1 = S$ and the S_{i+1} are obtained by applying a critical substitution to S_i or applying a single rewrite rule from S_i. Then one can derive a set S_k containing either an equation of the form $y = t$ or $t = y$ where the variable y does not appear in t, or an inequation $u \neq v$ where u and v are rigid unifiable, from S using a number of critical substitution applications and rewriting steps that is at most cubic in the subterm size of S. For this, we allow rigid rewrite operations as above, that may apply a rewrite rule $r \rightarrow s$ from S_i simultaneously to replace all occurrences of r in $S_i - \{r \rightarrow s\}$ by s.*

Proof. We construct a sequence $(S_1, \Theta_1), (S_2, \Theta_2), \cdots, (S_k, \Theta_k)$ where $S_1 = S$ and $\Theta_1 = \Theta$ and for all i, $S_i\Theta_i$ is ground and logically equivalent to $S\Theta$ (and is therefore unsatisfiable relative to the equality axioms). Furthermore, S_k contains an inequation $u \neq v$ for rigid unifiable u and v, or an equation of the form $y = t$ or $t = y$ where the variable y does not appear in t. Also, each S_{i+1} is obtained from S_i by a critical substitution application or a rigid rewriting operation, and k is cubic in the subterm size of S. By theorem 3.5, if S_i does not already contain an equation of the form $y = t$ or $t = y$ where y does not appear in t, we can assume that Θ_i is $S_i\Theta_i$-irreducible. If S_i does not already contain an inequation $u \neq v$ for u and v rigid unifiable, then either case 2 or 3 of theorem 3.13 holds. Whenever case 2 holds, we can perform all possible rigid rewriting steps by regarding the variables of S_i as constant symbols and using Method A or B above. For this, we order terms using the ordering $>_{\Theta_i}$ defined by $t >_{\Theta_i} u$ iff $t\Theta_i > u\Theta_i$. It is straightforward to verify that this is a termination ordering. Thus we initially have the set R of rules defined by $\{r \rightarrow s : r\Theta_i > s\Theta_i, r = s \in S_i \textit{ or } s = r \in S_i\}$. These rules may, however, have variables in s that do not appear in r. Processing this R using Method A or B, and also rewriting the inequation when possible, produces S_{i+1}, and we can let Θ_{i+1} be Θ_i. This processing takes a number of steps quadratic in the subterm size of S_i. This processing preserves logical equivalence of S_i and S_{i+1}, by repeated application of lemma 3.8. After this is done, case 3 of the theorem must hold. When this occurs, we can let S_{i+1} be $S_i\alpha$ where α is the critical substitution binding a variable, and let Θ_{i+1} be a substitution such that $S_i(\alpha \cdot \Theta_{i+1}) = S_i\Theta_i$, and continue. This is possible because $S_i\Theta_i$ is an instance of $S_i\alpha$. Now, logical equivalence is preserved because $S_{i+1}\Theta_{i+1} = S_i\Theta_i$. We note that $S_i\alpha$ contains at least one fewer variable than S_i. Therefore, the number of such critical substitution applications is at most linear in the subterm size of S, since the number of variables altogether is linear in the subterm size of S. Between the critical substitution applications, we may perform a quadratic number of rigid rewrite operations; the total number of steps is then cubic in the subterm size of S. It is also necessary to note that none of the rewriting or critical substitution applications increase the subterm size of S_i, that is, $|St(S_{i+1})| \leq |St(S_i)|$. This follows for the critical substitutions by lemma 3.11 and for the rewriting operations by lemma 2.21. □

We note that if S_i does contain such an equation $y = t$ or $t = y$, then from any inequation, we can derive an instance of $x \neq x$ in two (non-rigid) inference steps. For this, note that for an arbitrary inequation $s_1 \neq s_2$ we can replace y by s_1 and s_2, respectively, to obtain the instances $s_1 = t$ and $s_2 = t$, which contradict the inequation

$s_1 \neq s_2$; a two-step critical pair proof of some instance of $x \neq x$ can always be found. Since two different instances of y are used, this part of the proof is non-rigid. Therefore, we do not include it in the above corollary.

Definition 3.15 *We define $St_{max}(S)$ to be the maximal subterm size of any element of S.*

We note that $g = |S|$ (the cardinality of S), and the subterm size of S is bounded by $|S| * St_{max}(S)$. Thus we have a cubic bound on proof length (number of critical pair or rewrite operations) in terms of $|S| * St_{max}(S)$. Each rewrite operation of course is fairly powerful, but can be done efficiently if the proper data structures are used. The ground instances $S\Theta$ can be exponentially large in g and the size of S. However, since their subterm size is small, they can be represented by polynomial size directed acyclic graphs. We cannot give a polynomial time method of finding such proofs, because of complexity considerations. But we can show that such a proof exists, and of course if the ground instances are known, it can be constructed. It is somewhat surprising that the length of this proof does not depend on the ground instances T at all, just on the amplification S.

One may object that the proof "constructed" is short, but each operation may be very costly. This is because we may be performing critical pair operations on terms that are very large (exponential in the size of S). However, it follows from the above corollary that the subterm size of all terms generated in the proof will be bounded by $St(S)$, since $St(S_i) \leq St(S)$ for all i. Also, it is possible to unify two terms in time proportional to their subterm size. Therefore, each unification operation involved in the computation of needed critical pairs can be done in a time proportional to $St(S)$.

The proof operations are costly in another sense, as well. Namely, each rigid rewrite operation may apply to each element of S_i, and each critical pair operation may apply a substitution to each element of S_i. Therefore, we represent the proof as a sequence $S_1, S_2, S_3, \cdots$ of sets of equations and inequations, where each S_i is obtained from S_{i-1} by applying a rigid rewrite rule everywhere possible, or by applying a critical substitution. Then the length of this proof is cubic in $g * St_{max}(S)$. This is a reasonable way to represent the proof, since these operations on S_i may efficiently be done. However, when represented in a conventional way as a sequence of equations and inequations, each derived by a critical pair operation or a rewrite operation, then the length of the proof may be $O(g^4 St_{max}(S)^3)$, since S has g elements. But in this conventional non-rigid representation, substitutions no longer are applied to every element of S simultaneously. This can lead to exponentially long proofs, as the following example shows.

Example 3.16 Let A be the following (non-rigid) set of equations and one inequation.

$$f_1(x_1, x_2) \neq g_1(y_1, y_2)$$

$$f_1(f_2'(x_1, x_2), f_2'(y_1, y_2)) = h_1(f_2(x_1, x_2), f_2(y_1, y_2))$$

$$g_1(g'_2(x_1,x_2),g'_2(y_1,y_2)) = h_1(g_2(x_1,x_2),g_2(y_1,y_2))$$
$$f_2(f'_3(x_1,x_2),f'_3(y_1,y_2)) = h_2(f_3(x_1,x_2),f_3(y_1,y_2))$$
$$g_2(g'_3(x_1,x_2),g'_3(y_1,y_2)) = h_2(g_3(x_1,x_2),g_3(y_1,y_2))$$
$$\ldots$$
$$f_n(x_1,x_2) = h_n(x_1,x_2)$$
$$g_n(x_1,x_2) = h_n(x_1,x_2)$$

We first consider a rewriting proof of unsatisfiability from this example using the non-rigid framework, that is, variables in different equations can be renamed at will. Assuming a termination ordering $>$ so that all equations can be oriented left-to-right into rewrite rules, the only way to derive a contradiction using critical pair operations is to instantiate the left-hand side $f_1(x_1,x_2)$ to $f_1(f'_2(x_1,x_2),f'_2(y_1,y_2))$ and then rewrite it to $h_1(f_2(x_1,x_2),f_2(y_1,y_2))$, and similarly for the right-hand side, obtaining the inequation $h_1(f_2(x_1,x_2),f_2(y_1,y_2\ )) \neq h_1(g_2(z_1,z_2),g_2(w_1,w_2\ ))$. With more critical pair operations, this becomes $h_1(h_2(f_3(x_1,x_2),\ f_3(x_3,x_4)),\ h_2(f_3(y_1,y_2),$ $f_3(y_3,y_4))) \neq h_1(h_2(g_3(z_1,z_2),g_3(z_3,\ z_4)),\ h_2(g_3(w_1,w_2),g_3(w_3,w_4)))$. Eventually we construct terms of exponential size, and these are made unifiable by the equations $f_n(x_1,x_2) = h_n(x_1,x_2)$ and $g_n(x_1,x_2) = h_n(x_1,x_2)$. Thus this proof requires an exponential number of critical pair operations.

We now consider the rigid framework applied to this same example. Let S be an amplification of A in which each equation and inequation appears exactly once, but with distinct variables. This suffices, because in this example we only need one instance of each equation, for $2n+1$ instances in all, including an instance of the inequation. This makes g small. In this framework, the variables in all copies of the same equation are kept the same, so that instead of the equation given above we obtain $h_1(h_2(f_3(x_1,x_2),f_3(x_3,x_4)),$ $h_2(f_3(x_1,x_2),f_3(x_3,x_4))) \neq h_1(h_2(g_3(y_1,y_2),g_3(y_3,y_4)),\ h_2(g_3(y_1,y_2),g_3(y_3,y_4)))$ by critical pair operations. Thus the number of distinct subterms created in the rigid framework is small, and the terms can all be represented by polynomial size directed acyclic graphs. Furthermore, the proof can be found in a polynomial number of steps.

Since a realistic inference system may not use rigid operations, one might ask how to obtain polynomial proofs in cases such as this. One possibility of course is to add some rigid operations; indeed, this may be an advantage of using rigid operations as in Andrews' system [And81]. Another possibility is to allow critical pair operations that permit the overlap to occur on the small side of the equation; thus, in an equation $t = u$, if for all Θ such that $u\Theta$ and $t\Theta$ are ground, $u\Theta < t\Theta$, we could still unify a subterm of u with the left or right-hand side of some other equation. This permits a polynomial length proof of the above example to be found, but the equations derived have exponential subterm size; possibly there is some way to represent these terms efficiently, despite this. Still another possibility is to retain the restriction that critical pairs can only be done on the large side of an equation, but add the operation of unifying two subterms of a term, and saving the resulting instance of an equation or inequation. Thus we have the following proposed operation:

Definition 3.17 *The* subterm unification operation *on an equation or inequation $e[t,u]$ having unifiable subterms t and u, with $t \not\equiv u$, infers from e the instance $e\alpha$ of e, where α is a most general unifier of t and u.*

We note that the consideration of such specialized operations motivated by our complexity analysis is one of the benefits of this approach, since such techniques can conceivably improve the performance of term-rewriting based theorem provers. The soundness of this subterm unification operation follows from the fact that free variables are assumed to be universally quantified, so in fact any instance of e can be inferred. Thus we could unify the two subterms $h_2(f_3(x_1,x_2), f_3(x_3,x_4))$ and $h_2(f_3(y_1,y_2), f_3(y_3,y_4))$ of the term $h_1(h_2(f_3(x_1,x_2), f_3(x_3,x_4)),\ h_2(f_3\ (y_1,y_2),\ f_3(y_3,y_4)))$. However, this operation creates instances of more general equations, and such instances are usually deleted. Therefore, we propose to modify instance deletion as follows:

Definition 3.18 *The* large instance deletion operation *deletes a (non-trivial) instance $e\beta$ of an inferred equation or inequation e if $|St(e\beta)| \geq |St(e)|$ and $e\beta \not\equiv e$.*

If $e\alpha$ is inferred from e by subterm unification, then $|St(e\alpha)| < |St(e)|$, so $e\alpha$ would not be deleted. This unification of subterms operation is analogous to merging (unification) of literals in clauses in first-order theorem proving, and this large instance deletion operation is in fact analogous to the way the subsumption deletion in first-order resolution is often implemented, whereby a clause D can be deleted if there is a clause C and a substitution Θ such that $C\Theta \subseteq D$ and such that $|C| \leq |D|$. As pointed out by Christopher Lynch, it may also be useful to restrict rewriting. Otherwise, the instance $e\alpha$ may be deleted right away: if an equation $r = s$ with $r > s$ is instantiated to $r\alpha = s\alpha$, then $r\alpha$ will immediately rewrite to $s\alpha$, so this instance will rewrite to $s\alpha = s\alpha$ and be deleted. Therefore, we may disallow rewriting an equation e by a rewrite rule $r \rightarrow s$ if $|St(e)| \leq |St(r \rightarrow s)|$.

This subterm unification operation does permit a polynomial size proof to be found for example 3.16, and in general, as we now show.

Definition 3.19 *We say that the term, equation or inequation e' is a* coherent instance *of e if there is a substitution Θ such that $e\Theta$ is identical to e' and such that for any two subterms t and u of e, if $t\Theta \equiv u\Theta$ then $t \equiv u$.*

Lemma 3.20 *Suppose t' is a coherent instance of t and u' is a coherent instance of u. Suppose t' and u' are (rigidly) unifiable with most general unifier α'. Let α be a most general (non-rigid) unifier of t and u. Then there exists a term v obtainable from $t\alpha$ by a sequence of at most $|St(t) + St(u)|$ subterm unification steps such that $t'\alpha'$ is a coherent instance of v.*

Proof. We know that $|St(t\alpha)| \leq |St(t) \cup St(u)|$ by lemma 3.11, essentially. Now, each subterm unification operation on a term reduces its subterm size by one or more.

Also, if $t'\alpha'$ is not already a coherent instance of $t\alpha$, then it is possible to perform a subterm unification on $t\alpha$. Therefore, after a number of subterm unifications bounded by $|St(t\alpha)|$, either no more such unifications are possible, or else $t'\alpha'$ is a coherent instance of the term v obtained from $t\alpha$. We note that if no more subterm unifications are possible, then automatically $t'\alpha'$ is a coherent instance of the term v obtained from $t\alpha$. □

Theorem 3.21 *Suppose that A is a set of equations and an inequation such that A is unsatisfiable relative to the equality axioms. Let S be an amplification of A and let Θ be a ground substitution such that $S\Theta$ is ground and unsatisfiable relative to the equality axioms. (These must exist, by Herbrand's theorem.) Let us consider proofs as non-rigid sequences $e_1, e_2, \cdots, e_k$ of equations and inequations, where each e_i is either in A or is derived from previous e_j by a critical pair operation, a rewriting operation, or a subterm unification operation. Then one can derive an inequation $u \neq v$ for rigid unifiable u and v from A using a number of (non-rigid) critical pair operations, rewriting steps, and subterm unification operations that is $O(g^5 St_{max}(S)^4)$. For this, we allow parallel rewrite operations as above, that may apply a rewrite rule $r \rightarrow s$ simultaneously to replace all occurrences of $r\alpha$ in e_i by $s\alpha$, for arbitrary α. Also, we allow arbitrary renamings of variables in equations and inequations.*

Proof. Recall that the proof obtained in corollary 3.14 has a length cubic in $|St(S)|$, that is, in $g * St_{max}(S)$. We can simulate the rigid proof of corollary 3.14 by a non-rigid proof from A, with an extra factor of g coming from the fact that the elements of S_i have to be listed separately. We simulate the proof of corollary 3.14 by constructing a sequence of equations and inequations e_j having the elements of S_i as coherent instances; this permits corresponding rewrite or critical pair operations to be performed on the e_j. Also, extra subterm unification operations are needed to insure that subterms in the non-rigid e_j derivation from A are equal when the corresponding subterms in the rigid S_i derivation from S are. The number of these subterm unifications is bounded by the subterm size of S, by lemma 3.20. This adds another factor of $gSt_{max}(S)$. The proof from corollary 3.14 may derive an equation of the form $y = t$ or $t = y$ where the variable y does not appear in t; if so, then in two more non-rigid critical pair operations we can derive an inequation of the form $u \neq u$. This is noted after the proof of corollary 3.14. □

As in corollary 3.14, this bound on proof length does not depend on the ground instances T, but only on the amplification S. The following result helps to explain this lack of dependence on T, by showing that ground instances exist that are small in a certain sense. This result, and the one following it, seem to be related to results presented in [Gou94].

Theorem 3.22 *Suppose S is an unsatisfiable set of equations and inequations and Θ is a substitution such that $S\Theta$ is ground and unsatisfiable relative to the equality axioms. Then there is a substitution Θ' such that $S\Theta'$ is ground and unsatisfiable relative to equality and such that $|St(S\Theta')| \leq |St(S)|$.*

Proof. Consider the proof constructed in corollary 3.14. This involves a sequence of critical pair operations and rigid rewriting steps. Let α_i be the critical substitution applied to S_i, or, the identity if a rigid rewriting step was applied to S_i to obtain S_{i+1}. Consider the set $S\alpha_1\alpha_2 \cdots \alpha_{k-1}$. A similar proof of unsatisfiability can be constructed from this set; however, since the critical substitutions have all been applied at the start, the proof can be found purely by rigid rewriting steps. Let Θ' be $\alpha_1\alpha_2 \cdots \alpha_{k-1}\beta$ where β replaces all remaining variables by a fixed constant symbol; then $S\Theta'$ is ground and unsatisfiable relative to the equality axioms. Also, it follows by repeated application of lemma 3.11 that $|St(S\Theta')| \leq |St(S)|$. □

In fact, there is an analogous result for first-order logic without equality:

Theorem 3.23 *Suppose S is a set of clauses and Θ is a substitution such that $S\Theta$ is ground and unsatisfiable (not relative to equality). Then there is a substitution Θ' such that $S\Theta'$ is ground and unsatisfiable and such that $|St(S\Theta')| \leq |St(S)|$.*

Proof. We can obtain Θ' as the composition of a sequence of *matching* substitutions α_i, where a matching substitution is a most general unifier of literals L and M for literals L and M such that L appears in some clause of S and the complement of M appears in some clause in S. This follows from the completeness proof of clause linking, given in [LP92]. That proof was given in a non-rigid framework, but the idea directly transfers to a rigid framework as well. Reasoning as above, each such matching substitution α_i does not increase the subterm size of S. Also, it is necessary to apply some β at the end, replacing remaining variables by a constant symbol, in order to ensure that $S\Theta'$ is a set of ground clauses. □

However, we cannot obtain polynomial bounds on proof lengths for the general first-order case, even in terms of $|St(S)|$, at least not using known proof systems. We can obtain a bound that is exponential in $|St(S)|$ for the first-order case using instantiation by matching as in the theorem, followed by a propositional decision procedure. The number of matching steps needed to generate Θ' is linear in $|St(S)|$, since each such matching substitution eliminates a variable from S; also, propositional decision procedures often run fast in practice. For Horn clauses, this procedure would generate proofs of length polynomial in $|St(S)|$, since Horn sets can be decided quickly in the propositional case. Theoretically, we can obtain similar bounds using an equality transformation as in [Bra75] to eliminate the equality axioms and then apply the above theorem. This gives a general way to combine equality and first-order logic, albeit without traditional rewriting techniques. But it is still not clear from a theoretical standpoint what the proper way to combine equality reasoning with general first-order logic is. For some additional discussion of this problem, see [Pla94].

In general, we have the following bound on proof length, using a more powerful critical pair operation:

Definition 3.24 *We define the* parallel critical pair operation *which performs many disjoint critical pairs at the same time, that is, it unifies one side r of an equation*

$r = s$ with an arbitrary number of disjoint subterms of an equation or inequation e_i and then replaces all these subterms of e_i by corresponding instances of s.

Theorem 3.25 *Suppose that A is a set of equations and an inequation such that A is unsatisfiable relative to the equality axioms. Let S be an amplification of A and Θ a ground substitution such that $S\Theta$ is ground and unsatisfiable relative to the equality axioms. Let us consider proofs as (non-rigid) sequences $e_1, e_2, \cdots, e_k$ of equations and inequations, where each e_i is either in A or is derived from previous e_j by a non-rigid critical pair operation or a non-rigid rewriting operation. Suppose that we permit these proofs to contain the parallel critical pair operation just defined, but not the subterm unification operation. Then one can derive an inequation $u \neq v$ for rigid unifiable terms u and v, from A using a number of critical pair operations and rewriting steps that is $O(g^4 St_{max}(S)^3)$. For this, we allow rewrite operations as above, that may apply a rewrite rule $r \rightarrow s$ simultaneously to replace all occurrences of r in e_i by s.*

Proof. We can obtain this by lifting the proof obtained in corollary 3.14. Corresponding to each element e'_j of a set S_j in the rigid proof from S there constructed, we have equations and inequations e_i having e'_j as an instance in a non-rigid proof from A. We then can perform non-rigid rewriting operations and critical pair operations similar to those of the proof of corollary 3.14, on the more general terms. The equations e_i may be more general than e'_j, because critical substitutions are only applied to the involved equation, and not to other elements of S_j. We need the parallel critical pair operation to be able to lift the rewriting operation of corollary 3.14, since the rewriting operation of corollary 3.14 can replace an arbitrary number of terms at the same time. These terms are all identical in the rigid proof from S constructed in 3.14, but in the more general non-rigid proof from A constructed here, they may be distinct, so one may need many critical pair operations. The length of the proof is $O(g^4 St_{max}(S)^3)$, because S has g elements. We note that in the proof of 3.14, we may derive an equation of the form $y = t$ or $t = y$ where the variable y does not appear in t. If so, we can in two critical pair steps derive an inequation of the form $u \neq u$, as noted in the comments after corollary 3.14. □

Corollary 3.26 *Suppose that A is a set of equations and an inequation such that A is unsatisfiable relative to equality. Let S be an amplification of A and let Θ be a ground substitution such that $S\Theta$ is ground and unsatisfiable relative to the equality axioms. Let us consider proofs from A as non-rigid sequences $e_1, e_2, \cdots, e_k$ of equations and inequations, where each e_i is either in A or is derived from previous e_j by a non-rigid critical pair operation or a non-rigid rewriting operation. For this, we only allow the usual rewrite operation, which may apply a rewrite rule $r \rightarrow s$ to replace one occurrence of $r\alpha$ in e_i by $s\alpha$. Suppose also that we permit these proofs to contain ordinary critical pair operations, but not the parallel critical pair operation nor the subterm unification operation. Then one can derive an inequation $u \neq v$ for rigid unifiable u and v from A using a number of non-rigid critical pair operations and rewriting steps that is $O(g^4 St_{max}(S)^3 * a^{|St(S)|})$, where a is the maximum arity of any function symbol in S.*

Proof. Let n be $|St(S)|$. Then the depth of any term t in S is at most n. Therefore the size (number of subterm occurrences) of t is at most a^n, where a is the maximum arity of any function symbol in S. To each operation in the non-rigid proof from A constructed in theorem 3.25, we may have a^n operations in the non-rigid proof from A constructed with less powerful rewriting and critical pair operations. This is true because $|St(S_i)| \leq |St(S)|$ for all i in corollary 3.14, and because for all equations e_j in the proof from A constructed in theorem 3.25, there exists an i such that e_j has some element of S_i as an instance. This implies that the depth of e_j is at most $|St(S_i)|$. $\square$

Though this bound is exponential, it does at least depend only on S and not on the ground instances T. This result also gives us some evidence that the more powerful operations make a significant difference. We now consider a restricted case in which a smaller bound on proof length can be derived.

Definition 3.27 *If E is a set of equations, we write $r_1 \leftrightarrow_E r_2$ if there is some equation $t_1 = t_2$ or $t_2 = t_1$ in E and r_1 may be expressed as $r_1[t_1\Theta]$ and r_2 may be expressed as $r_1[t_2\Theta]$ for some substitution Θ. Thus r_2 can be obtained from r_1 by using some equation of E in a forwards or backwards direction.*

Definition 3.28 *An* equational proof *of an equation $s_1 = s_2$ from a set E of equations is a sequence $r_0, r_1, \ldots, r_n$ of terms, where r_0 is s_1 and r_n is s_2 and for each i, $r_i \leftrightarrow_E r_{i+1}$. The* length *of this proof is n.*

Birkhoff showed that $E \models s_1 = s_2$ (for ground s_1, s_2) iff there is an equational proof of $s_1 = s_2$ from E.

Theorem 3.29 *Suppose there is an equational proof of some instance $s_1\alpha = s_2\alpha$ of the equation $s_1 = s_2$ from a set E of equations, and this proof has length n. Then there is a proof of some inequation $u \neq v$ where u and v are rigid unifiable from $E \cup \{s_1 \neq s_2\}$ using a number of non-rigid critical pair and rewriting operations which has at most $O(n^4 * St_{max}(E \cup \{s_1 \neq s_2\})^3)$ steps.*

Proof. The number g of ground instances of equations used in the equational proof is bounded by $n+1$. By corollary 3.14, there is a critical pair-rewriting proof of some inequation $u \neq v$ where u and v are rigid unifiable, whose length is $O(n^4 * St_{max}(E \cup \{s_1 \neq s_2\})^3)$. $\square$

It is possible to derive a tighter bound in case E is left and right-linear.

Definition 3.30 *A term is* linear *if it has at most one occurrence of each variable. An equation $r = s$ is* linear *if both r and s are linear. We also call such an equation left and right-linear, for emphasis.*

Theorem 3.31 *Suppose there is an equational proof of some ground instance $s_1\alpha = s_2\alpha$ of the linear equation $s_1 = s_2$ from a set E of linear equations, and this proof has length n. Then there is a non-rigid critical pair proof of some inequation $u \neq v$ where u and v are rigid unifiable from $E \cup \{s_1 \neq s_2\}$ which involves at most n critical pair operations.* (*Note that no rewriting operations are needed.*)

Proof. Let A be $E \cup \{s_1 \neq s_2\}$. It is clear that A is unsatisfiable relative to equality. Let $r_0, r_1, \ldots, r_n$ be an equational proof of $s_1\alpha = s_2\alpha$, where $r_0, r_1, \ldots, r_n$ are ground terms. Let $t_i = u_i$ be the rewrite rule used to obtain r_i from r_{i-1}, and let $t_i\Theta_i = u_i\Theta_i$ be the instance of this rule that is used to obtain r_i from r_{i-1}. Let e_i be the equation $t_i = u_i$. We transform the equational proof of $s_1\alpha = s_2\alpha$ by a series of proof transformation steps. Let r_i be the maximal (with respect to $>$) term in this derivation. We assume that this term r_i is larger than its two neighboring terms; if not, it must be an endpoint of the derivation, or else two adjacent terms r_i and r_{i+1} are identical. In the latter case, one of these terms can be omitted, shortening the derivation. If this term appears somewhere in the middle of the sequence, and is larger than r_{i-1} and r_{i+1}, it is called a *peak*. Then we may be able to form a (non-rigid) critical pair between the equations e_i and e_{i+1}, obtaining an equation that can be used to obtain r_{i+1} from r_{i-1} by one replacement step. This reduces the length of the equational proof by one. Otherwise, it can be that these successive replacement steps occur at independent positions, in which case we can reorder them, obtaining one or two (or perhaps no) smaller peaks. We call this a *rearrangement* of the proof. Otherwise, one replacement step is performed on a term that occurs within a subterm $x\Theta_i$ or $x\Theta_{i+1}$ for some variable x in e_i or e_{i+1}. Suppose the equation $t_i = u_i$ used to obtain r_i from r_{i-1} can be expressed as $t_i[x] = u_i[x]$, where we indicate the occurrences of one of the variables x in this manner. Let us express r_{i-1} as $r_i[t_i\Theta_i[w[t_{i+1}\Theta_{i+1}]]]$ and r_i as $r_i[u_i\Theta_i[w[t_{i+1}\Theta_{i+1}]]]$ and r_{i+1} as $r_i[u_i\Theta_i[w[u_{i+1}\Theta_{i+1}]]]$. Since $r_i > r_{i-1}$ and $r_i > r_{i+1}$, we have $t_i\Theta_i < u_i\Theta_i$ and $t_{i+1}\Theta_{i+1} > u_{i+1}\Theta_{i+1}$. Then we can interchange these steps so that the replacement using $t_{i+1} = u_{i+1}$ is done first, obtaining r_i' as $r_i[t_i\Theta_i[w[u_{i+1}\Theta_{i+1}]]]$. Now $r_i' < r_{i-1}$ and $r_i' < r_{i+1}$. We can then consider the new derivation in which r_i is replaced by r_i'. We also call this proof transformation a rearrangement. This rearrangement reduces the size of r_i, and therefore of the multiset of r_i, relative to the ordering $<$. This corresponds to changing the order of application of e_i and e_{i+1}, and will not increase the length of the equational proof. If E were not linear, such a transformation could still be done, but it might increase the length of the equational proof, since x may have more than one occurrence on one or both sides of the equation $t_i = u_i$. The situation if r_i occurs at an endpoint of the sequence is similar, but may involve a critical pair or rearrangement operation involving the inequation $s_1 \neq s_2$. Such a critical pair or rearrangement operation will be possible as long as this maximal term is larger than one of its neighboring terms. The only other possibility is that the derivation is of the form r_0, r_1 where $r_0 \equiv r_1$, in which case we have already derived an inequation $u \neq v$ where u and v are rigid unifiable. By continuing this process, eventually we obtain a critical pair proof having length n or less. The base case is a proof of the form r_0, r_1 where $r_0 \not\equiv r_1$; from this we can derive an inequation $u \neq v$ for u and v rigid unifiable in one critical pair operation involving the inequation $s_1 \neq s_2$. $\square$

Unfortunately, these proof length bounds may be difficult to achieve in practice. This is because other rewrite rules may be derived in an implementation of completion, and these unnecessary rules may cause reductions to occur that are not part of the desired proof. The implication of this is that in order to guarantee that a polynomial size proof can be found, as stated in the above results, it may be necessary to restrict the rewriting operation in ways that may be hard to determine in advance. Of course, one possibility is to always save both the original form of every clause as well as its rewritten form. This guarantees that a short (polynomial in g etc.) proof will eventually be found, if the search strategy is fair. But this has obvious disadvantages in terms of the search space size. Another possibility is to show that applying rewrite rules of a certain form can never eliminate all polynomial size proofs. For example, rewrites of the form $r \rightarrow s$, where s is a proper subterm of r, or where s is a constant symbol, cannot eliminate short proofs, assuming the search strategy is fair, and assuming that these rewriting steps are not counted in the proof length. Such rewrites never increase $|St(S)|$ by more than the number of constant symbols, and so a polynomial bound on proof length is maintained.

4 Acknowledgements

The comments of Christopher Lynch were very helpful in the writing of this paper.

References

[And81] P. B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28:193–214, 1981.

[BDP89] Leo Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In Hassan Ait-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures 2: Rewriting Techniques*, pages 1–30, New York, 1989. Academic Press.

[BO84] G. Bauer and F. Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica*, 21:521–540, 1984.

[Bra75] D. Brand. Proving theorems with the modification method. *SIAM J. Comput.*, 4:412–430, 1975.

[CL73] C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[CMO93] D.E. Cohen, K. Madlener, and F. Otto. Separating the intrinsic complexity and the derivational complexity of the word problem for finitely presented groups. *Mathematical Logic Quarterly*, 39:143–157, 1993.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, Amsterdam, 1990.

[GNP+93] Jean Gallier, P. Narendran, D. Plaisted, S. Raatz, and W. Snyder. An algorithm for finding canonical sets of ground rewrite rules in polynomial time. *J.ACM*, 40:1:1 – 16, 1993.

[GNPS90] J. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. *Information and Computation*, 87(1/2):129–195, July/August 1990. Special issue devoted to LICS '88.

[GNRS92] J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *J. ACM*, 39(2):377–429, 1992.

[Gou94] Jean Goubault. The complexity of resource-bounded first-order classical logic. In P. Enjalbert, E.W. Mayr, and K.W. Wagner, editors, *11th Symposium on Theoretical Aspects of Computer Science*, pages 59–70, Caen, France, February 1994. Springer Verlag LNCS 775.

[Klo92] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1 – 117. Oxford University Press, Oxford, 1992.

[Lov78] D. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, New York, 1978.

[LP92] S.-J. Lee and D. Plaisted. Eliminating duplication with the hyper-linking strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[Lyn95] C. A. Lynch. Paramodulation without duplication. In *Proceedings 10th IEEE Symposium on Logic in Computer Science, San Diego*, June 1995.

[MO85] K. Madlener and F. Otto. Pseudo-natural algorithms for the word problem for finitely presented monoids and groups. *Journal of Symbolic Computation*, 1:383–418, 1985.

[MSKO93] K. Madlener, A. Sattler-Klein, and F. Otto. On the problem of generating small convergent systems. *Journal of Symbolic Computation*, 16(2), 1993.

[Pla93] D. Plaisted. Equational reasoning and term rewriting systems. In D. Gabbay, C. Hogger, J. A. Robinson, and J. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 273–364. Oxford University Press, 1993.

[Pla94] D. Plaisted. The search efficiency of theorem proving strategies: an analytical comparison. Technical Report MPI-I-94-233, Max-Planck Institut fuer Informatik, Saarbruecken, Germany, 1994.

[Sny89] W. Snyder. Efficient ground completion: an $O(nlogn)$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E. In *Proceedings of the 3rd International Conference on rewriting techniques and applications*, pages 419–433, 1989. Lecture Notes in Computer Science, Vol. 355.

[WOLB84] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning: Introduction and Applications.* Prentice Hall, Englewood Cliffs, N.J., 1984.