A New Method of Discovering Mathematical Proofs—Research Paper

Matthew Weidner
Contents

1 Introduction 2

2 The Method 3
  2.1 The Method of General Matings 3
  2.2 This Project’s Method 5

3 Proof Lengths for The Method 6

4 The Search Space of The Method 10

5 An Instance of The Method 12

6 Adding Equality 14
  6.1 Adding Theories, Generally 14
  6.2 A Rigid Proof Procedure for Equality 15
  6.3 The Method Plus Equality 18

7 Discussion 19

8 Conclusions & Future Work 20
1 Introduction

Automated theorem proving is an area of computer science dedicated to automatically producing and verifying mathematical proofs. It has applications in many fields, including mathematics and software verification. For a brief and accessible overview, see [25]. This project exposites and analyzes a method of automated theorem proving for first-order logic, a logical language which is widely used in the automated theorem proving area.

Usually, automated theorem provers construct proofs step-by-step. Starting with the negation of the theorem being proven, new formulas are deduced and added to the set of proven facts. This process is repeated until a contradiction is derived. By tracing the path of deductions that lead from the input to the eventual contradiction, a human-style proof is constructed.

This project presents a method of theorem proving based on proof discovery instead of the usual proof construction. The method is an apparently original instance of a more general method called the method of general matings [1]. The method of general matings reduces the theorem proving problem to the problem of finding an object called a refutation mating. Existing instances of the method of general matings construct refutation matings step-by-step, while this project’s method searches for them in a certain natural search space.

The method also distinguishes itself from existing methods in its use of external tools. Algorithms implementing the method will be able to export most of their work to unification algorithms and SAT solvers (decision procedures for propositional/boolean logic), both of which have been intensely researched and made rather efficient. The effects of this are threefold. First, the method is simplified because it only has to focus on one aspect of the theorem proving problem. Second, the efficiency of the external tools will hopefully lead to the efficiency of implementations of the method. Third, as the external tools become more efficient, so will the method.

There are two classes of existing methods that use SAT solvers. One comprises the instance-based methods, which use resolution-like inference rules to generate ground instances of clauses. These clauses are added to a set of already-generated ground instances and checked for unsatisfiability using a SAT solver. The other is the method of ChewTPTP, which encodes the existence of a rigid proof as a boolean satisfiability problem and solves that problem using a SAT solver. The instance-based methods use proof construction instead of proof discovery, making them distinct from this project’s method. ChewTPTP uses a SAT solver for a different purpose.
than this project’s method.

There also is an existing instance of the method of general matings. It appears that this instance, first presented in [1], is the only existing instance of the method of general matings. It is not compatible with SAT solvers and relies on proof construction instead of discovery, making it distinct from this project’s method.

Besides presenting the method, this project analyzes its theoretical efficiency by comparing its proof lengths to those of other methods. The search space associated with the method is discussed, and powerful ways to guide the search are developed. In particular, an algorithm based on the method is written, and lower bounds on its efficiency are established.

The initial method works in the language of first-order logic without equality, since this is the language of the method of general matings. However, this project also presents a version of the method that works for first-order logic with equality. In addition, theoretical results are established which should allow the method to be extended to other theories.

The outline of this paper is as follows. Section 2 gives a brief overview of the method of general matings before presenting this project’s method. Section 3 compares proof lengths in the method to proof lengths in the resolution method, the rigid resolution method, and the method used by ChewTPTP. Section 4 discusses the size and structure of the method’s search space. Section 5 uses the search space’s structure to create an algorithm based on the method, and it establishes lower bounds on this algorithm’s efficiency. Section 6 extends the method to handle equality. Section 7 discusses the results of this project. The final section contains conclusions and future work.

2 The Method

In this section, this project’s new method of automated theorem proving is developed, and its correctness is proven. We work in the language of first-order logic without equality.

2.1 The Method of General Matings

First, we introduce the necessary parts of the method of general matings. A thorough description of the method, including proofs, is found in [1].

A negation normal form formula (nnf) is a formula in which each occurrence of ̄ occurs directly in front of an atom. A negation normal form
A universal nns is a nns which contains no occurrences of $\exists$. Using De Morgan’s laws and Skolemization, one can turn any formula into a universal nns which is unsatisfiable iff the original formula is unsatisfiable. Because a formula is valid iff the negation of its universal closure is unsatisfiable, the problem of showing that a given first-order formula is valid can be reduced to the problem of showing that a certain universal nns is unsatisfiable. It is this latter problem that we focus on.

A truth assignment is a function from atoms to truth values ($\{\top, \bot\}$). Given a truth function $T$ and a quantifier-free formula $\varphi$, the value of $\varphi$ under $T$ is defined in a natural way. A quantifier-free formula is said to be truth-functionally contradictory (t-f contradictory) if its value is false under all truth functions.

A quantifier-free nns $\varphi$ is said to be rigidly unsatisfiable if there exists a substitution $\sigma$ such that $\varphi\sigma$ is t-f contradictory.

A quantifier duplication of a formula $\varphi$ is any process which consists of replacing a subformula of $\varphi$ of the form $\forall x \psi$ with the formula $(\forall x \psi) \land (\forall x \psi)$. The normalization of a formula $\varphi$ is the formula which results from replacing each bound variable with a new variable (to avoid name conflicts) and then removing all quantifiers. An amplification of a universal nns $\varphi$ is any result of duplicating any number of quantifiers in $\varphi$ and then normalizing. For example, suppose $f$ is a unary function symbol, $c$ and $d$ are constant symbols, $x$ and $y$ are variables, $P$ is a binary relation symbol, and $Q$ and $R$ are unary relation symbols. Then one of the amplifications of $\forall x (P(x, y) \land (\forall y (Q(y) \lor R(x))))$ is the formula $P(x', z) \land (Q(y') \land R(x')) \land (Q(v) \land R(x'))$.

Although it is not stated explicitly, the next lemma follows easily from results proven in [1].

**Lemma 2.1.** Let $\varphi$ be a universal nns. Then $\varphi$ is unsatisfiable iff there is an amplification $\psi$ of $\varphi$ such that $\psi$ is rigidly unsatisfiable.

Thus, the first-order unsatisfiability problem can be reduced to a (usually infinite) series of rigid unsatisfiability problems.

To solve rigid unsatisfiability problems, matings are used. Let $\varphi$ be a quantifier-free nns, and let $A$ be the set of all atoms occurring in $\varphi$. A potential mating of $\varphi$ is an equivalence relation on $A$ such that each non-singleton equivalence class contains at least one atom occurring positively in $\varphi$ and at least one atom occurring negatively in $\varphi$. Note that an atom may occur both positively and negatively in $\varphi$. If $\sim_M$ is a potential mating of $\varphi$ and there exists a substitution $\sigma$ such that, for all $\alpha, \beta \in A$, $\alpha \sim_M \beta$ implies
\[ \alpha \sigma = \beta \sigma, \] then \( \sim_M \) is unifiable and is a mating of \( \varphi \). If \( \sim_M \) is a potential mating of \( \varphi \) such that \( \varphi \) is false under every truth assignment which respects \( \sim_M \) (i.e., assigns equal values to atoms in the same equivalence class under \( \sim_M \)), then \( \sim_M \) is unsatisfiable and is a refutation potential mating of \( \varphi \). A potential mating of \( \varphi \) which is both unifiable and unsatisfiable is a refutation mating of \( \varphi \). See Figure 1.

<table>
<thead>
<tr>
<th>Unifiable?</th>
<th>No/ Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Refutation mating</td>
</tr>
<tr>
<td></td>
<td>Refutation potential mating</td>
</tr>
<tr>
<td>No/ Unknown</td>
<td>Mating</td>
</tr>
<tr>
<td></td>
<td>Potential mating</td>
</tr>
</tbody>
</table>

Figure 1: The Types of Matings

The key to the method is the following lemma.

**Lemma 2.2.** Let \( \varphi \) be a quantifier-free nnf. Then \( \varphi \) has a refutation mating iff \( \varphi \) is rigidly unsatisfiable.

Combining the two lemmas gives:

**Theorem 2.3.** Let \( \varphi \) be a universal nns. Then \( \varphi \) is unsatisfiable iff some amplification of \( \varphi \) has a refutation mating.

### 2.2 This Project’s Method

We now develop this project’s method as an instance of the method of general matings.

The main idea of the new method is to search for refutation matings in the space of all potential matings by enumerating potential matings, checking each one for unifiability and unsatisfiability. This is more of a suggestion than a truly new method, but it is apparently unexplored and is worth exploring. The new method, in its most general form, does not specify how potential matings are enumerated. However, helpful results are given in Section 4. Additionally, an example instance of the method which does specify how potential matings are enumerated is given in Section 5.

5
It is apparent that the unifiability of a potential mating is equivalent to the unifiability of any set of equations between atoms whose symmetric-reflexive-transitive closure is the potential mating. Thus, the unifiability property can be checked using any unification algorithm.

With regards to unsatisfiability, define the *propositionalization* of a formula $\varphi$ under a potential mating $\sim_M$ as the propositional formula which results from replacing each atom $\alpha$ in $\varphi$ with a propositional variable $P[\alpha]$ corresponding to the equivalence class of $\alpha$ under $\sim_M$. We then have the following easy theorem.

**Theorem 2.4.** Let $\sim_M$ be a potential mating of $\varphi$. Then the propositionalization of $\varphi$ under $\sim_M$ is unsatisfiable iff $\sim_M$ is unsatisfiable.

Thus, because SAT solvers decide propositional satisfiability, the unsatisfiability of a potential mating can be tested by running a SAT solver on the corresponding propositionalization.

### 3 Proof Lengths for The Method

In this section, the efficiency of the method is analyzed by looking at its shortest proof lengths. Its proof lengths are compared to those of resolution, rigid resolution, and a proof-based version of ChewTPTP. Also, the method’s place in the polynomial-time hierarchy is determined.

First, a proof-based version of the method is defined. For simplicity, assume that all inputs are given as quantifier-free sets of clauses. The definition of quantifier duplications for sets of clauses is modified by defining a *quantifier duplication* of a clause set $C$ as any process which results in a set $C \cup \{c'\}$, where $c'$ is the result of renaming the variables of a clause $c \in C$.

Define a *proof line* as an ordered 4-tuple $(\varphi, \sim_M, m, r)$, where $\varphi$ is a set of clauses, $\sim_M$ is NULL or a potential mating, $m$ is UNK or TRUE, and $r$ is UNK or TRUE. A *reputation* of the set of clauses $c$ in the method is a series of proof lines such that: the first proof line is $(C, NULL, UNK, UNK)$; each proof line is the output of one of the below-listed inference rules applied to an earlier proof line; and the final proof line has TRUE as its last two elements.

The inference rules of the proof-based version of the method are:

- **Duplicate**: Input $(C, NULL, UNK, UNK)$. Output $(C', NULL, UNK, UNK)$, where $C'$ is the result of a quantifier duplication applied to $C$. 

6
• **Mate**: Input $(C, NULL, UNK, UNK)$. Output $(D, \sim_M, UNK, UNK)$, where $\sim_M$ is a potential mating for $C$ and $D$ is the propositionalization of $C$ under $\sim_M$.

• **Check_unifiable**: Input $(C, \sim_M, UNK, r)$, where $\sim_M$ is unifiable. Output $(C, \sim_M, TRUE, r)$.

• **Check_unsatisfiable**: Input $(C, \sim_M, m, UNK)$, where $\sim_M$ is not NULL and $C$ is t-f contradictory. Output $(C, \sim_M, m, TRUE)$.

By Theorem 2.3, a set of clauses is unsatisfiable iff it has a refutation in the method.

This proof system is not a proof system in the ordinary sense. However, except for rule **Check_unsatisfiable**, it is a proof system in the more technical sense that its proofs can be verified in time polynomial in the size of the proof.

The proof system of the method is compared with that of ChewTPTP. ChewTPTP can be thought of as a proof system with two inference rules, one of which adds a copy of each input clause to the current formula, and the other of which checks rigid unsatisfiability by calling a SAT solver on a problem of size $O(n^3)$, where $n$ is the length of the current formula. Corresponding to each application of the first inference rule, use $k$ applications of **Duplicate**, where $k$ is the number of clauses in the input. Corresponding to the second inference rule, use one application each of **Mate**, **Check_unifiable**, and **Check_unsatisfiable**. Thus, the method’s proofs are of size linear in the size of those of ChewTPTP, and both ChewTPTP and the method use one call to a SAT solver. However, the method calls the SAT solver on a problem of size $O(n)$ while ChewTPTP does on a problem of size $O(n^3)$. This should correspond to a drastic efficiency improvement on the part of the method. Note, though, that ChewTPTP has practically no search space while the method has a large one.

For the purpose of fair comparison with resolution and rigid resolution, **Check_unsatisfiable** needs to be modified so that its uses can be verified in polynomial time. To this end, create the amended proof system FAIR by replacing **Check_unsatisfiable** with the propositional resolution rules:

• **Prop_resolve**: Input $(C, \sim_M, m, UNK)$, where $\sim_M$ is not NULL. Output $(C \cup \{e\}, \sim_M, m, UNK)$, where $e$ is the result of applying the propositional resolution rule to some $c, d \in C$. 

7
• **Prop\_factor**: Input \((C, \sim_M, m, UNK)\), where \(\sim_M\) is not NULL. Output \((C \cup \{d\}, \sim_M, m, UNK)\), where \(d\) is the result of applying the propositional factoring rule to some \(c \in C\).

• **Find\_unsatisfiable**: Input \((C, \sim_M, m, UNK)\), where \(\sim_M\) is not NULL and \(C\) contains the empty clause. Output \((C, \sim_M, m, TRUE)\).

**Theorem 3.1.** Let the length of a proof line be given by its size when written as a string. Then each line of a proof in FAIR can be checked in time polynomial in the length of the line.

**Proof.**

• **Duplicate**: Trivial.

• **Mate**: First, we must check that the domain of \(\sim_M\) is exactly the set of atoms of \(C\). Next, we must check that each non-singleton equivalence class contains at least one atom occurring positively in \(\phi\) and at least one atom occurring negatively in \(\phi\). Finally, we must construct the propositionalization of \(C\) under \(\sim_M\). Each of these operations can be done in time linear in the size of the proof line.

• **Check\_unifiable**: Let \(n\) be the size of \(\sim_M\). The unifiability of \(\sim_M\) corresponds to the unifiability of a certain easily-found set of at most \(\frac{n(n-1)}{2}\) equations, each of which equates two atoms of \(C\). Thus, we must solve a unification problem of size quadratic in the size of \(C\). Because there exist polynomial-time unification algorithms such as Baxter’s algorithm [5], this proves the claim.

• **Prop\_resolve, Prop\_factor**: Well-known.

• **Find\_unsatisfiable**: Trivial.

Assume that we use SAT solvers that are generally the best available, and thus are hopefully as fast as first-order theorem prover that uses resolution when applied to a propositional problem. Then one can say, in a sense, that the proofs of the method are at least as “short” as those of FAIR, where “short” means a more vague measure of length than straightforward proof length. Thus, proof lengths in FAIR are a good estimate of “proof lengths” (in the vague sense) for the method.

Because of this, we now compare the lengths of proofs in FAIR to the lengths of proofs in resolution and rigid resolution.
Theorem 3.2. Let $C$ be an unsatisfiable set of clauses that have shortest rigid resolution proof of length $n$. Then there is a refutation of $C$ in FAIR of length $O(n)$.

Proof. Construct a refutation in FAIR as follows. Start with $(C, NULL, UNK, UNK)$. This is rigidly unsatisfiable by hypothesis, so it has a refutation mating $\sim_M$. Apply the instance of Mate which sets the second element of the line to $\sim_M$. Then apply Check_unifiable. This takes 2 proof lines.

Next, apply Prop_resolve and Prop_factor so as to simulate the rigid resolution proof, by applying them to the propositional versions of the clauses that there first-order counterparts are applied to in the rigid resolution proof. The most recent line now has first element which contains the empty clause, so we end by applying Find_unsatisfiable. This completes the refutation. It takes $n + 1$ proof lines.

Thus, there exists a refutation in FAIR of length $n+3 \in O(n)$, as claimed. \[\square\]

Theorem 3.3. There exists an infinite sequence of unsatisfiable formulas $\varphi_1, \varphi_2, \ldots$ whose shortest FAIR proof has length exponential in the length of the shortest resolution proof.

Proof. By Theorem 8.3.2 of [22], there exists an infinite sequence of unsatisfiable clause sets $C_1, C_2, \ldots$ such that $C_n$ has a resolution proof of length $\lceil \log_2(n) \rceil + 2$, but $n - 1$ quantifier duplications must be applied to $C$ before it becomes rigidly unsatisfiable. Specifically, let $a$ be a constant symbol, $f$ be a unary function symbol, $x$ be a variable, and $P$ be a unary predicate symbol. Then we define

$$C_n = \{ P(a), \neg P(x) \lor P(f(x)), \neg P(f^n(a)) \}$$

where $f^n$ denotes $n$-fold application of $f$. \[\square\]

By inspecting the proof of Theorem 3.2, we see that this difference in proof length can only appear as the result of a formula’s requiring many quantifier duplications before becoming rigidly unsatisfiable. Thus, on formulas which do not require many quantifier duplications, resolution’s advantage disappears.

As a side effect of Theorem 3.1, we have the following result.

Theorem 3.4. The method, when applied to a rigid unsatisfiability problem, can run on a nondeterministic polynomial-time Turing machine equipped with an oracle for the boolean satisfiability problem.
Because rigid unsatisfiability is $\Sigma^p_2$-complete [18], this is the lowest that any rigid unsatisfiability solver can be in the polynomial-time hierarchy (unless the hierarchy collapses).

4 The Search Space of The Method

Even when we restrict ourselves to the rigid unsatisfiability problem, the method leaves one major step undefined: choosing potential matings. In this section, we discuss the search space associated with this choice.

One could create a complete instance of the method by having the mating step merely enumerate all possible potential matings of $\psi$. However, this will be highly inefficient: letting $B_n$ be the $n$th Bell number, there are $B_n$ possible equivalence relations on the atoms of $\psi$, most of which are potential matings. $B_n$ grows at least as fast as $(\frac{n}{e\ln n})^n$ asymptotically [7]. It is conceivable that $\psi$ could be rigidly unsatisfiable even if only one of its potential matings is a refutation mating, and finding one refutation mating amongst nearly $B_n$ potential matings will require more than blind guessing.

It turns out that the search space has structure that we should be able to exploit, allowing us to avoid blind guessing.

First, expand the search space slightly to give it more structure: instead of considering only potential matings, consider the set of all potential near-matings, which are defined as the equivalence relations on the set of atoms of $\psi$. The terminology of potential matings applies to near-matings in a natural way.

This does not harm the correctness of the method, as the following theorem shows.

**Theorem 4.1.** Let $\varphi$ be a universal nns. Then $\varphi$ is unsatisfiable iff some amplification of $\varphi$ has a refutation near-mating.

*Proof.* ($\Rightarrow$): By Theorem 2.3, if $\varphi$ is unsatisfiable, then some amplification of it has a refutation mating. Because every refutation mating is a refutation near-mating, this proves the claim.

($\Leftarrow$): By Lemma 2.1, we must show that, if a quantifier-free nnf $\psi$ has a refutation near-mating, then $\psi$ is rigidly unsatisfiable. This follows from the definitions of refutation near-mating and rigid unsatisfiability.

It is reasonable to suspect that most potential near-matings are potential matings, so expanding the search space in this way should not increase its size by an intolerable amount.
Now, the structure. By thinking of the potential near-matings as partitions of the set of atoms, we can impose a natural structure on the set by partially ordering its elements by partition refinement. In this way, we form a geometric lattice.¹ We will use \( \sim_M \leq \sim_N \) to denote that \( \sim_M \) is a refinement of \( \sim_N \).

The following results outline the use of \( \leq \).

**Theorem 4.2.** Let \( \sim_M \) and \( \sim_N \) be potential (near-)matings of \( \psi \) such that \( \sim_M \leq \sim_N \). Then (i) if \( \sim_M \) is a refutation potential (near-)mating, then so is \( \sim_N \), and (ii) if \( \sim_N \) is a (near-)mating, then so is \( \sim_M \).²

**Proof.** For each cell \( a \) of a potential (near-)mating, let \( P_a \) denote the propositional variable corresponding to \( a \) in the propositionalization of \( \psi \) under it. To prove part (i), we show the contrapositive. To this end, assume that \( \sim_N \) is not a refutation potential (near-)mating. Then the propositionalization of \( \psi \) under \( \sim_N \) is satisfied by some model; call it \( w \). Let \( w' \) be given by: for each cell \( c \) of \( \sim_M \), \( w'(P_c) = w(P_d) \), where \( d \) is the cell of \( \sim_N \) that is a superset of \( c \). Then \( w' \) is a model of \( \sim_M \), so \( \sim_M \) is also not a refutation potential (near-)mating.

To prove part (ii), note that the unification problem corresponding to \( \sim_N \), considered as a set of equations, is a superset of the unification problem corresponding to \( \sim_M \). Thus, the unifiability of the unification problem for \( \sim_N \) implies the unifiability of the unification problem for \( \sim_M \). \( \square \)

**Corollary 4.3.** Say \( \psi \) has a refutation (near-)mating. Let \( \sim_M \) be a potential (near-)mating of \( \psi \) which is not a refutation (near-)mating.

(i) If \( \sim_M \) is a (near-)mating but not a refutation potential (near-)mating, then there is a refutation (near-)mating of \( \psi \) which is not finer than \( \sim_M \).

(ii) If \( \sim_M \) is a refutation potential (near-)mating but not a (near-)mating, then there is a refutation (near-)mating of \( \psi \) which is not coarser than \( \sim_M \).

(iii) If \( \sim_M \) is neither a (near-)mating nor a refutation potential (near-)mating, then there is a refutation (near-)mating of \( \psi \) which is incomparable with \( \sim_M \).

Thus, whenever a potential (near-)mating \( \sim_M \) fails either the unifiability check or the unsatisfiability check, we can remove an entire set of potential matings.

¹If we consider only the potential matings, then we still form a partial order, but it need not be graded or a lattice.

²Part (i) for potential matings has been noted already in [19], although the use that this project sees for it has not been previously recognized.
(near-)matings from the search space without compromising our ability to find a refutation (near-)mating. This set is usually quite large: let $\sim_M$ have $k$ cells of size $c_1, c_2, \ldots, c_k$. Then there are $B_k - 1$ near-matings that are strictly coarser than $\sim_M$, and there are $\prod_{i=1}^k B_i - 1$ near-matings that are strictly finer than $\sim_M$.

These results seem to have much potential for suggesting instances of the method. They may even enable us to detect rigid satisfiability, not just unsatisfiability, without having to check most of the search space. For instance, there is at least one way to find sets of potential near-matings with the same number of cells such that, if all elements of some set are nonunifiable or satisfiable (unifiable or unsatisfiable), then we can remove from the search space all potential-near matings with more (less) cells.

Specifically, we have the following simple theorem. An integer partition of a positive integer $n$ is a multiset of positive integers which sum to $n$, and the shape of a set partition $p$ is the integer partition whose elements are the sizes of the cells of $p$.

**Theorem 4.4.** Let $X$ be a set of size $n$. Let $\lambda$ be an integer partition of $n$ into $k$ parts, and suppose that $\lambda$ is a refinement of every integer partition of $n$ into $k - 1$ parts. Then every partition of $X$ with less than or equal to $k - 1$ parts is coarser than some element of $\{p | p$ is a partition of $X$ that has shape $\lambda\}$.

An analogous result holds for $\geq$.

This idea is made more promising by the fact that there already exist software packages for finding and manipulating these kinds of sets of partitions [20].

## 5 An Instance of The Method

This section presents an application of Lemma 4.2. Specifically, the application is an instance of the method that is based on finding maximal unifiable subsets of a certain constraint set.

Let $\leq_m$ be the restriction of $\leq$ to the set of all matings of $\psi$. Use the term maximal mating to describe any mating which is maximal with respect to $\leq_m$. Then we find the following theorem.

**Theorem 5.1.** If $\psi$ has a refutation mating, then it has a maximal mating which is a refutation mating.
Proof. Let $\sim_M$ be a refutation mating of $\varphi$. Because $\sim_M$ is a mating and so is in the set that is partially ordered by $\leq_m$, there is a maximal mating $\sim_N$ such that $\sim_M \leq \sim_N$. By Lemma 4.2, $\sim_N$ is a refutation potential mating, and so it is a refutation mating.

Thus, without sacrificing completeness, we can enumerate only the maximal matings of $\psi$.

Finding all maximal matings of $\psi$ is a problem in itself, but it turns out that it can be solved by an existing algorithm. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be the atoms of $\psi$, and let $C$ be the set of unification constraints $\{\alpha_1 = \alpha_2, \alpha_1 = \alpha_3, \ldots, \alpha_1 = \alpha_n, \alpha_2 = \alpha_3, \alpha_2 = \alpha_4, \ldots, \alpha_{n-1} = \alpha_n\}$. Then the maximal matings of $\psi$ correspond to the maximal unifiable subsets of $C$ in a natural way, where a maximal unifiable subset is a unifiable subset of $C$ which is not a proper subset of any unifiable subset of $C$. An algorithm for computing all maximal unifiable subsets of a constraint set has been developed by Cox [10].

Although the instance of the method has a smaller search space than the method in general, the search space may still be of factorial size, as the next theorem shows.

**Theorem 5.2.** Suppose we implement the instance of the method by choosing random maximal matings without replacement and checking each for unsatisfiability. Then there exists an infinite sequence $\varphi_1, \varphi_2, \ldots$ of unsatisfiable formulas of increasing size on which we expect (in the sense of expected value) to enumerate $\frac{1}{2} \left( \frac{|\varphi_n|}{2} \right)!$ maximal matings before finding a refutation mating.

Proof. Let $c_1, c_2, \ldots$ be constant symbols, let $x_1, x_2, \ldots$ be variables, let $f$ be a unary function symbol, and let $P$ be a unary predicate symbol. Define $\varphi_n$ by

$$\varphi_n = (P(c_1) \lor P(f(c_2)) \lor \ldots \lor P(f^{n-1}(c_n))) \land \neg P(v_1) \land \neg P(f(v_2)) \land \ldots \land \neg P(f^{n-1}(v_n))$$

where $f^m$ denotes $m$-fold application of $f$.

The maximal matings correspond to ways of choosing one element of $\{P(v_1) = P(c_1), P(v_1) = P(f(c_2)), \ldots, P(v_1) = P(f^{n-1}(c_n))\}$, one element of $\{P(f(v_2)) = P(f(c_2)), P(f(v_2)) = P(f(f(c_2))), \ldots, P(f(v_2)) = P(f^{n-1}(c_n))\}$, etc. Thus, there are $n \times (n-1) \times \ldots \times 1 = n!$ maximal matings. However, only one of these is unsatisfiable, namely, the one corresponding to $P(v_1) = P(c_1), P(f(v_2)) = P(f(c_2)), \ldots, P(f^{n-1}(v_n)) = P(f^{n-1}(c_n))$. 

13
Thus, we expect to enumerate $\frac{n!}{2} = \frac{1}{2} \left(\frac{|\text{free}(n)|}{2}\right)!$ maximal matings before finding a refutation mating.

\[\square\]

6 Adding Equality

The method developed so far handles first-order logic without equality. However, many more interesting and practical problems could be tackled if the method could handle equality. To this end, a version of the method for first-order logic with equality is presented in this section.

6.1 Adding Theories, Generally

First, we prove theoretical results that partially generalize the method to work modulo first-order theories. While the original method reduces the unsatisfiability problem (modulo the empty theory) to a series of ground unsatisfiability problems (modulo the empty theory), the results of this section reduce the unsatisfiability problem modulo an arbitrary theory $T$ to a series of ground unsatisfiability problems modulo $T$.

Some notation is needed. For a quantifier-free formula $\varphi$ and integer $k \geq 1$, $\varphi^k$ denotes $(\varphi_{\rho_1} \land \varphi_{\rho_2} \land \ldots \land \varphi_{\rho_k})$, where each $\rho_i$ is a distinct renaming substitution. $\text{free}$ is a function mapping an expression to the set of all free variables occurring in that expression. $g$ is an arbitrary fixed substitution that maps each variable to a new constant. It is used to “ground” terms, turning variables into constants.

Let $T$ be a first-order theory. Define a rigid proof procedure for $T$ as a function $\text{rpp}$ from quantifier-free formulas to sets of sets of unification constraints such that, for all formulas $\varphi$,

(i) every $E \in \text{rpp}(\varphi)$ is unifiable

(ii) every $E \in \text{rpp}(\varphi)$ is finite

(iii) for all substitutions $\sigma$, if $\sigma$ unifies some $E \in \text{rpp}(\varphi)$, then $\varphi\sigma$ is unsatisfiable modulo $T$

(iv) if $\varphi$ is unsatisfiable modulo $T$, then for some $k$, $\text{rpp}(\varphi^k) \neq \emptyset$

Properties (ii) and (iii) correspond to soundness and completeness, respectively. Note that $\varphi$ is unsatisfiable iff $\text{rpp}(\varphi^k) \neq \emptyset$ for some $k$. 
Given a rigid proof procedure \( rpp \), define a covering function for \( rpp \) as a function \( E \) from quantifier-free formulas to sets of equations such that, for all \( \varphi \), \( \bigcup rpp(\varphi) \subset E(\varphi) \).

**Theorem 6.1.** Let \( T \) be a first-order theory, let \( rpp \) be a rigid proof procedure for \( T \), and let \( E \) be a covering function for \( rpp \). Then a quantifier-free formula \( \varphi \) is unsatisfiable modulo \( T \) iff there is a \( k \) and a unifiable set of equations \( E \subset E(\varphi^k) \) such that \( \varphi^k mgu(E)g \) is unsatisfiable modulo \( T \).

**Proof.** Trivial. \( \square \)

Some more notation is needed. Define a partition \( p \) of terms to be unifiable if some substitution unifies all cells of \( p \). A side of an equation \( s \approx t \) is either \( s \) or \( t \). Note that equations are implicitly unoriented, i.e., \( s \approx t \) is indistinguishable from \( t \approx s \).

**Corollary 6.2.** Let \( T \) be a first-order theory, let \( rpp \) be a rigid proof procedure for \( T \), let \( E \) be a covering function for \( rpp \), and let \( U(\varphi) \) be the set of all sides of equations in \( E(\varphi) \). Then a quantifier-free formula \( \varphi \) is unsatisfiable modulo \( T \) iff there is a \( k \) and a unifiable partition \( p \) of \( U(\varphi^k) \) such that \( \varphi^k mgu(p)g \) is unsatisfiable modulo \( T \).

**Corollary 6.3.** Let \( T \) be a first-order theory, let \( rpp \) be a rigid proof procedure for \( T \), let \( E \) be a covering function for \( rpp \), and let \( U(\varphi) \) be the set of all sides of equations in \( E(\varphi) \). Furthermore, suppose \( S_1(\varphi), S_2(\varphi), \ldots, S_n(\varphi) \) are sets of terms such that, for all \( e \in E(\varphi) \) and indices \( i \), some side of \( e \) is in \( S_i \). Then a quantifier-free formula \( \varphi \) is unsatisfiable modulo \( T \) iff there is a \( k \) and a unifiable partition \( p \) of \( U(\varphi^k) \) such that (i) \( \varphi^k mgu(p)g \) is unsatisfiable modulo \( T \), and (ii) each non-singleton cell of \( p \) contains at least one element of \( S_i \) for all indices \( i \).

Of course, not all theories will have computable rigid proof procedures, and not all rigid proof procedures will have covering functions with finite outputs.

### 6.2 A Rigid Proof Procedure for Equality

However, the theory of equality does have a rigid proof procedure. Specifically, a rigid proof procedure is naturally suggested by the *strict basic superposition calculus* \([4]\).

First, some preliminaries. The notation \( t[u] \) is ambiguously used to denote a term \( t \) with subterm \( u \). If \( t[u] \) has been written, then a future writing
of \( t[v] \) ambiguously denotes the result of replacing a single occurrence of \( u \) in \( t \) with \( v \). The strict basic superposition calculus inputs clauses where the only predicate is the equality predicate \( \approx \). This is not a restriction, since an arbitrary atom \( P(t_1, t_2, \ldots, t_n) \) can be presented in a two-sorted first-order language using an equality \( P(t_1, t_2, \ldots, t_n) \approx \top \), where \( \top \) is a distinguished constant of sort \( \text{bool} \). Negative literals \( \neg(s \approx t) \) are written \( s \not\approx t \). Positive literals are called \emph{equations} and negative literals are called \emph{disequations}. We assume a fixed reduction ordering\(^3\) \( \prec \) which is total on ground terms.

The inference rules of the strict basic superposition calculus operate on \emph{closures}, which consist of a clause \( C \) and a constraint \( E \) and are written \( C \cdot E \). The constraint is a set containing both \emph{unification constraints} of the form \( s = t \) and \emph{inequality constraints} of the form \( s \prec t \). A substitution \( \sigma \) satisfies a constraint \( E \) if it unifies each unification constraint in \( E \) and, for all inequality constraints \( (s \prec t) \in E \), \( s\sigma \prec t\sigma \) holds. A constraint is \emph{satisfiable} if some substitution satisfies it and \emph{unsatisfiable} otherwise.

The \emph{rigid strict basic superposition calculus} RSBS consists of the following inference rules. It is implicitly assumed that the constraint on the output of each rule is satisfiable.

\begin{itemize}
  \item \textbf{Positive strict basic superposition}

\[ \frac{(C \lor s \approx t) \cdot E \cdot (D \lor w[s'] \approx v) \cdot F}{(w[t] \approx v \lor C \lor D) \cdot \{ s = s', s \succ t, w[s'] \succ v \} \cup E \cup F} \]

where \( s' \) is not a variable.

\item \textbf{Negative strict basic superposition}

\[ \frac{(C \lor s \approx t) \cdot E \cdot (D \lor w[s'] \not\approx v) \cdot F}{(w[t] \not\approx v \lor C \lor D) \cdot \{ s = s', s \succ t, w[s'] \succ v \} \cup E \cup F} \]

where \( s' \) is not a variable.

\item \textbf{Reflexivity resolution}

\[ \frac{(s \not\approx s' \lor C) \cdot E}{C \cdot \{ s = s' \} \cup E} \]

\item \textbf{Factoring}

\[ \frac{(s \not\approx s' \lor C) \cdot E}{C \cdot \{ s = s' \} \cup E} \]

\end{itemize}

\(^3\)A well-founded partial order \( \prec \) on terms such that, if \( a \prec b \), then \( t[a\sigma] \prec t[b\sigma] \) for all terms \( t \) and substitutions \( \sigma \).
\[
(s \approx t \lor s' \approx t' \lor C) \cdot E
\]

\[
(s \approx t \lor C) \cdot \{s = s', t = t'\} \cup E
\]

where \(s_\rho = s'\) and \(t_\rho = t'\) for some renaming substitution \(\rho\).

For a finite set of clauses \(S\), define \(rpp\)(\(S\)) by the set of all equation sets \(E\) such that \(\bot \cdot E \cup F\) can be derived in \(RSBS\) from \(\{C \cdot \emptyset | C \in S\}\), where \(F\) is any set of inequality constraints. For a quantifier-free negation normal form formula \(\varphi\), define \(rpp\)(\(\varphi\)) by \(rpp\)(\(S_\varphi\)), where \(S_\varphi\) denotes the clause form of \(\varphi\) obtained using the boolean distributive laws.

\(rpp\) has property (i) of rigid proof procedures by the definition of \(RSBS\). Because the ground version of \(RSBS\) without inequality constraints is sound, one can use a refutation of \(\varphi\) in \(RSBS\) ending in \(\bot \cdot E \cup F\) to construct a refutation of \(\varphi\)\(\text{mgu}(E)\)\(g\). Thus, \(rpp\) has property (iii). Also, it is proven in [4] that, if \(S\) is a set of clauses in first-order logic with equality, then \(S\) is unsatisfiable iff for some \(k\), there exists a derivation of \(\bot \cdot G\) from \(\{C \cdot \emptyset | C \in S^k\}\) in \(RSBS\). Thus, \(rpp\) has property (iv).

It remains to be seen that every element of every output of \(rpp\) is finite. To this end, let \(URSBS\) denote the proof calculus consisting of the restriction of \(RSBS\) to unit clauses.

**Lemma 6.4.** Let \(S\) be a set of clauses in first-order logic with equality, and let \(L\) be the set of all literals occurring in \(S\). Then

(i) if a term \(t\) appears in a (possibly incomplete) \(RSBS\) derivation from \(S\), then \(t\) appears in an \(URSBS\) derivation from \(L\)

(ii) if a term \(t\) appears as a side of an equation in an \(RSBS\) derivation from \(S\), then \(t\) appears as a side of an equation in an \(URSBS\) derivation from \(L\)

**Proof.** For each inference rule in \(RSBS\) that creates a literal, the corresponding inference rule in \(URSBS\) applied to the same input literals creates the same literal. The claim then follows from well-founded induction on the length of the \(RSBS\) derivation.

**Lemma 6.5.** Let \(L\) be a finite set of equations and disequations. Then only finitely many terms appear in all \(URSBS\) derivations from \(L\).

**Proof.** By Theorem 7 of [12], there are only finitely many derivations in \(URSBS\) from \(L\), each of which contains only finitely many terms.
Collecting results, we have

**Theorem 6.6.** \( rpp_\tau \) is a rigid proof procedure for first-order logic modulo equality.

### 6.3 The Method Plus Equality

With respect to a given formula \( \varphi \), define a partition \( E \) of terms to be **unsatisfiable** if \( (\varphi \land E)g \) is unsatisfiable modulo equality.

We are now ready to prove the main theorem of this project’s method plus equality.

**Theorem 6.7.** Let \( \varphi \) be a quantifier-free negation normal form formula in first-order logic modulo equality. Let \( \mathcal{U}(\varphi^k) \) denote the set of all terms that occur in UR\( SBS \) derivations from literals occurring in \( \varphi^k \), and let \( \mathcal{V}(\varphi^k) \) denote the set of all terms that occur as sides of equations in UR\( SBS \) derivations from literals occurring in \( \varphi^k \). Then \( \varphi \) is unsatisfiable modulo equality iff there is a \( k \) and a partition \( p \) of \( \mathcal{U}(\varphi^k) \) such that (i) \( p \) is unifiable, (ii) \( p \) is unsatisfiable, and (iii) each non-singleton cell of \( p \) contains at least element of \( \mathcal{V}(\varphi^k) \).

**Proof.** (\( \Rightarrow \)): Note that every unification constraint generated by RSBS is of the form (side of equation = term). Then by Corollary 6.3, there is a \( k \) and a unifiable partition \( p \) of \( \mathcal{U}(\varphi^k) \) such that \( p \) satisfies (i) and (iii) and \( \varphi^kmgu(p)g \) is unsatisfiable modulo equality. Note, though, that the proof that \( rpp_\tau \) has property (iii) of rigid proof procedures only requires that the toplevel terms in \( p \) become equal for a ground proof to exist. That is, \( \varphi g \) can be proved unsatisfiable using \( pg \) as a set of unit equations, even though \( (\varphi \land p)g \) is a slightly weaker statement than \( \varphi mgu(p)g \).

(\( \Leftarrow \)): By Corollary 6.3. \( \Box \)

Ground unsatisfiability problems can be solved using satisfiability modulo theories (SMT) solvers for the theory of equality. SMT solvers correspond to SAT solvers modulo equality.

The search space for the method plus equality has the same structure as the search space for the method without equality, as the next theorem shows.

**Theorem 6.8.** Let \( p \) and \( q \) be partitions of \( \mathcal{U}(\varphi) \) such that \( p \) is finer than \( q \). Then (i) if \( p \) is unsatisfiable, then so is \( q \), and (ii) if \( q \) is unifiable, then so is \( p \).
The proof is very similar to the proof of the corresponding result for the method without equality (Theorem 4.2) and is left as an exercise for the reader.

As a final note, observe that Theorem 6.7 reduces to a slightly weakened form of Theorem 2.3 if \( \varphi \) is a formula without equality, i.e., a formula where every equation and disequation is of sort \( \text{bool} \). This is because we can ignore any partitions whose non-singleton cells contain terms not of sort \( \text{bool} \), since they would violate (iii) or be nonunifiable. Thus, \( \varphi \) is unsatisfiable iff some partition \( p \) of the atoms of \( \varphi \) satisfies (i) \( p \) is unifiable, (ii) \( p \) is unsatisfiable, and (iii) each non-singleton cell contains at least one positively-occurring and one negatively-occurring literal.

7 Discussion

The results about proof lengths indicate that the equality-less method has the potential to be efficient if equipped with good heuristics. Of course, shorter proof lengths do not always lead to better implementations, since shorter proof lengths correspond to larger search spaces, for which one is less likely to find good heuristics.

However, the results about the structure of the search space open up many possibilities. These results organize the search for proofs in a new way. Common proof methods, usually based on resolution, ultimately focus on deriving consequences of existing proven formulas. The method instead searches through a geometric lattice for an entire proof at once, without regard for individual formulas and without building up a set of consequences. In fact, because of its emphasis on delegating logical tasks to external tools, the method could conceivably have implementations which do not even acknowledge the fact that they are working with logical statements! Such implementations would instead focus solely on geometric and combinatorial aspects of the search space.

The one instance of the method, the algorithm based on maximal unifiable subsets, appears to be too inefficient to be practical. It may require factorially many calls to a SAT solver to solve rigid unsatisfiability problems. Note, though, that all of these calls will be on fairly “small” problems which are only as large as the formula being shown rigidly unsatisfiable. Additionally, it is open to improvement with heuristic methods, since it only has to find one successful element of the search space. The same can be said for the method in general.
The extension of the method to equality needs further development. The set of terms which must be partitioned, though finite, appears to be quite large. Indeed, it seems to be on the order of $s^n$ or larger, where $s$ is the size of the largest term and $n$ is the total number of terms appearing in $\varphi^k$. Nonetheless, by forming a tradeoff between $k$ and partition size, smaller sets-to-partition become feasible at the expense of increased $k$. Besides equality, the general results about theories open up the possibility of using the method on first-order logic modulo other theories.

The extension of the method to equality induces an extension of the method of general matings to equality. It appears that this is the first correct extension of this type. An extension was published in [15], but it is incorrect because it claims to decide the simultaneous rigid e-unification problem, which is undecidable [11].

8 Conclusions & Future Work

A new method of automated theorem proving for first-order logic without equality was presented. It stresses proof discovery instead of proof construction. It exports most of its work to external unification algorithms and SAT solvers, potentially giving it efficiency.

The method's proof lengths were compared to those of other methods, showing that it should be competitive against rigid resolution and ChewTPTP in general and against resolution on certain kinds of problems. A way to guide the search for refutation matings was presented, and from it, an algorithm implementing the method was developed. However, lower bounds on the runtime of the algorithm show that it is inefficient in general.

Some approaches to automated theorem proving rely on SAT solvers, which are computer programs that solve the boolean satisfiability problem. By rephrasing complex tasks as boolean satisfiability problems, one can execute them with ease. In addition, SAT solvers are efficient and constantly becoming better, and it is not unreasonable to believe that these qualities would carry over to programs that use SAT solvers.

There are two main existing approaches to automated theorem proving that use SAT solvers. One is comprised of the instance-based methods, which enumerate carefully-chosen sets of ground instances of the formula being proven. (Unsatisfiability of ground formulas can be tested using SAT solvers.) The sets are chosen such that the formula being proven is unsatisfiable iff one of the enumerated sets is unsatisfiable. Instance-based
methods include hyper-linking [23], Inst-Gen resolution [17], and partial-instantiations [21].

The second approach involves encoding the existence of proofs as boolean satisfiability problems. One method of this type is used by the recently developed automated theorem prover ChewTPTP [14]. For a given first-formula, ChewTPTP creates a boolean satisfiability problem that is usually satisfiable iff there exists a rigid tableau proof that the input formula is unsatisfiable.

The method presented by this project could be considered an instance-based method. To the author’s knowledge, though, it is distinct from all existing instance-based methods. This is especially due to its emphasis on discovering proofs in an abstract proof instead of making derivations using resolution-like rules.

The method was extended to reason about theorems involving equality. General results about using the method to reason modulo arbitrary theories were presented. In conjunction with results about strict basic superposition and rigid basic superposition, it was shown how to use partitions of certain sets of terms to extend the method. The extended method appears to be inefficient because the sets of terms seem large. The extended method reduces to the equality-less method when used on theorems that do not involve equality.

The method plus equality shows some similarity to instance-based methods plus equality [16]. In particular, both use sets of terms generated by unit rigid basic superposition to guide the generation of ground problems, which are then input to SMT solvers. Unlike instance-based methods plus equality, though, this project’s method plus equality stresses search and the partition lattice, while instance-based methods stress derivations.

For future work, implementations of the maximal matings algorithm should be developed. The results about the structure of the search space should be elaborated on further. New instances of the method should be developed. In addition, the existing instance should be improved. Heuristics should be investigated, both for the method in general and for its instances.

With regards to theories, the general results about using the method with theories should be expanded to work with theories besides equality. The size of term partitions in the method plus equality should be investigated and, if possible, reduced. Specific algorithms based on the method plus equality should be developed. In particular, the maximal matings algorithm should be extended to equality.

There are numerous practical applications of automated theorem provers,
in fields such as mathematics, software verification, hardware verification, and engineering [25]. In addition, there are uses for rigid unsatisfiability solvers, in fields such as cryptography [13].

References


