Path-Focused Duplication: A Search Procedure for General Matings

Sunil Issar
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890
si@cs.cmu.edu

Abstract

The mating paradigm for automated theorem provers was proposed by Andrews to avoid some of the shortcomings in resolution. It facilitates automated deduction in higher-order and non-classical logics. Moreover, there are procedures which translate back and forth between refutations by the mating method and proofs in a natural deduction system.

We describe a search procedure, called *path-focused duplication*, for finding refutations by the mating method. This procedure, which is a complete strategy for the mating method, addresses two crucial issues (inadequately handled in current implementations) that arise in the search for refutations: when and how to expand the search space. It focuses on a particular path that seems to cause an impasse in the search and expands the search space relative to this path in a way that allows the search to immediately resolve the impasse. The search space grows and shrinks dynamically to respond to the requirements that have arisen or have been met in the search process, thus avoiding an explosion in the size of the search space. We have implemented a prototype of this procedure and have been able to easily solve many problems that an earlier program found difficult.

Introduction

Much research in theorem proving has focused on improving the efficiency of procedures based on Robinson's resolution principle [Robinson, 1965]. The *mating* paradigm for automated theorem provers was proposed by Andrews [1981] (and a similar approach called the *connection* method was suggested by Bibel [1982]) to avoid converting a well-formed formula (wff) to clause form, which introduces redundancy and impedes analysis of the logical structure of the wff. As remarked by Stickel [1986], solutions to really hard problems will always require human assistance in specifying strategies and determining where to search for a solution; the mating method provides an attractive framework for this interaction because it retains the logical structure of the wff, and thus it is easier to see what the search is trying to achieve. According to Bibel [1988], the connection method has another advantage: *it is computationally as adequate as resolution and has the potential to do significantly better than resolution.*

This work was supported by NSF grant CCR-8702699.

The mating paradigm facilitates automated deductions in higher-order logic [Andrews, 1989] and non-classical logics [Wallen, 1990]. Refutations generated by the mating method are the underlying component of the proof presentation procedures in [Andrews, 1980; Miller, 1987; Pfenning, 1987; Pfenning and Nesmith, 1990]. These procedures translate back and forth between refutations by the mating method and proofs in a natural deduction system; the two systems can thus serve as cooperating processes, possibly in a semi-interactive environment, with each one of them trying to exploit the advantages of the other.

In this paper we focus on two crucial issues that arise in the search for refutations by the mating method: when and how to expand the search space. The procedure *path-focused duplication*, which is described later, addresses these issues. It focuses on a particular *path* that seems to cause an impasse in the search and expands the search space relative to this path; this transforms a global problem—how to expand the search space so that the search can succeed in finding a refutation—to a more specific local problem—which quantifier should we duplicate so that a particular impasse can be resolved.

As discussed in [Andrews, 1981], we cannot specify a bound on the size of the search space without sacrificing completeness, and each expansion increases the complexity of the search. (Bibel and his colleagues [Bibel *et al.*, 1983] do provide bounds for some special cases, but they also note that such bounds do not exist in general, since first-order logic is undecidable.) Search procedures for the mating method [Andrews, 1981; Bibel, 1982] have been based on the level-saturation approach [Chang and Lee, 1973]: exhaustively search for a *refutation* in a given wff before generating the expanded wff at the next level. These levels introduce artificial constraints in the search. The search procedures thus suffer from problems similar to the ones that arise in the depth-first approach, for example, the Horizon Effect [Berliner, 1973]. Since these problems are caused by the level-saturation approach, we rectify them by incorporating the expansion of the wff intrinsically into the search process. Path-focused duplication achieves this objective by eagerly *duplicating* an expansion node whenever the search seems to have reached an impasse. This could result in extremely large wffs, but we avoid this explosion in the search space by *localizing* the duplication to the impasse that caused it. Localization is achieved by a dynamically growing and shrinking wff which responds to the requirements that have arisen or have been met in the search process.

Just as model elimination [Loveland, 1978] and set of support [Wos *et al.*, 1965] are complete strategies for resolution, path-focused duplication is a complete strategy for the mating method. Path-focused duplication facilitates the introduction of many features (such as not looking for refutations in which wffs occur more than once without sacrificing completeness) which are mentioned in [Stickel, 1982] into the mating method.

The Mating Method

We start with a brief review of the mating method [Andrews, 1981; Bibel, 1982]. As is customary in theorem proving, we will remove existential quantifiers by Skolemization. The wff can contain any connectives, but we will transform them to disjunctions and conjunctions during translation to jforms (defined below). Since the same literal (an atomic wff or the negation of an atomic wff) can occur at several places in the wff, we will assign a unique label to each occurrence of a literal in the jform. A wff W is *rectified* iff no variable occurs both bound and free in W and distinct quantifier-occurrences of W bind distinct variables. In this paper we will deal only with the wffs that are rectified. This restriction is not a drawback of our procedures; we consider the problem of how to implicitly rectify a wff to be an important implementation detail, which is adequately addressed in [Issar, 1990]. $l_1@l_2@\ldots@l_n$ is the result of concatenating the lists $l_1, l_2, \ldots, l_n$. $[n]$ denotes the set $\{i \mid 1 \leq i \leq n\}$ of positive integers.

A *jform* (junctive form) is a representation as an and/or tree for a wff of first-order logic and is defined by the following BNF expression:

$$jform = Literal \mid \bigvee_{i=1}^{n} jform_i \mid \bigwedge_{i=1}^{n} jform_i \mid <\forall x_1 x_2 \ldots x_n> jform$$

We will refer to jforms of the form $<\forall x_1 x_2 \ldots x_n>$ *jform* as *expansion nodes* and refer to $x_1, x_2, \ldots, x_n$ as the *expansion variables* at this node. The expansion variables in any jform J are the expansion variables that occur at expansion nodes in J. The terminology expansion nodes and expansion variables was introduced by Miller [1987]. We will refer to expansion nodes by the expansion variables at that node. For example, $\forall uv$ refers to the expansion node $<\forall uv>$ $\neg Quv \vee Quu$ in Figure 1. Jforms are displayed as two-dimensional diagrams called *vpforms* (vertical path forms). In a vpform the disjunctions are displayed horizontally and the conjunctions are displayed vertically.

If x is an expansion variable in a jform J, then $J[y|x]$ is the result of replacing all occurrences of x by y in J. Let M be an expansion node in J, $\{x_i \mid i \in [n]\}$ be the set of expansion variables in the subtree with root node M, $\{y_i \mid i \in [n]\}$ be a set of new variables that do not occur in J, and M' be the result of replacing x_i by y_i in M for each $i \in [n]$; then M' is a *copy* of M. Consider the jform J' that is obtained from J by replacing M with the jform $\bigwedge M M'$. We say that J' is obtained by a *duplication* from J; this duplication is almost the same as *quantifier duplication* in [Andrews, 1981]. We also say that we *duplicated* M in J and denote J' by $J + M'$. J^* is an *amplification* of a jform J iff there exists a sequence $J = J_1, J_2, \ldots, J_{m-1}, J_m = J^*$ of jforms such that J_{i+1} is obtained by a duplication from J_i for each $i \in [m-1]$.

We next illustrate these definitions with a few examples:

Example 1 $\neg[\ [\ Rab \wedge \forall x \forall y[Rxy \supset [Ryx \wedge Qxy]]$
$\wedge \forall u \forall v[Quv \supset Quu]]$
$\supset\ [Qaa \wedge Qbb]]$

The jform J_1, which is displayed as a vpform, in Figure 1 represents this wff. The labels in this jform refer to the literals that are under them. $\forall x^1 y^1$ is a copy of the expansion node $\forall xy$. J_2 is an amplification of J_1.

Example 2 $\neg[[Pa \vee Pb \vee Pc] \supset \exists x Px]$

The jform J_3 in Figure 2 represents this wff. J_4 is an amplification of J_3.

The set Paths(J) of *paths* through a jform J is defined inductively as follows:

1. $J = Literal$. Then $Paths(J) \overset{\text{def}}{=} \{<J>\}$.
2. $J = \bigvee_{i=1}^{n} jform_i$. Then $Paths(J) \overset{\text{def}}{=} \bigcup_{i=1}^{n} Paths(jform_i)$.
3. $J = \bigwedge_{i=1}^{n} jform_i$.
   Then $Paths(J) \overset{\text{def}}{=} \{P_1@P_2@\ldots@P_n \mid P_i \in Paths(J_i)\}$.
4. $J = <\forall x_1 \ldots x_n>$ *jform*.
   Then $Paths(J) \overset{\text{def}}{=} \{<J>@P \mid P \in Paths(jform)\}$.

A *path* through a jform J is an element of the set Paths(J). We will sometimes consider a path to be an ordered set. A *partial path* through a jform J is a subset of a path through J. Our paths differ slightly from the paths in [Andrews, 1981] in that we add expansion nodes to the paths. We could as in [Pfenning, 1987] consider a path to be an element of the set of solution graphs [Nilsson, 1980] in J, but we are mainly interested in literals and expansion nodes, and thus restrict the paths to contain only these elements. To continue with our examples:

1. $<L_0, \forall xy, L_1, \forall uv, L_5, L_6>$ is a path through J_1.
2. For $i \in \{8, 9, 10\}$, let $P_i = <L_i, \forall x, L_{11}>$. Then Paths($J_3$) $= \{P_8, P_9, P_{10}\}$.
3. Paths(J_4) $= \{P@\ <\forall x_1, L_{11}^1, \forall x_2, L_{11}^2> \mid P \in Paths(J_3)\}$.

A substitution ρ is *more general* or *less specified* than a substitution ψ, denoted by $\rho \leq \psi$, iff there is a substitution ϕ such that $\phi\rho = \psi$. The relation $\leq$ is transitive. A substitution ρ is a *unifier* of a set $\mathcal{U}$ of pairs of literals iff $(t_1, t_2) \in \mathcal{U} \Rightarrow \rho t_1 = \rho t_2$. We will say that $\mathcal{U}$ is *unifiable* iff it has a unifier. ρ is the (essentially unique) *most general unifier* of $\mathcal{U}$ iff ρ is a unifier of $\mathcal{U}$ and $\rho \leq \alpha$ for any unifier α of $\mathcal{U}$.

A *connection* in a jform J is any unordered pair (L, M) of literals in J which are both on some path in J and have the property that the set $\{(\neg L, M)\}$ is unifiable. For ex-

$$J_1 = \begin{bmatrix} \overset{L_0}{Rab} \\ \forall xy \\ \begin{bmatrix} \overset{L_1}{\neg Rxy} \vee \begin{bmatrix} \overset{L_2}{Ryx} \\ \\ \overset{L_3}{Qxy} \end{bmatrix} \end{bmatrix} \\ \forall uv \\ \begin{bmatrix} \overset{L_4}{\neg Quv} \vee \overset{L_5}{Quu} \end{bmatrix} \\ \begin{bmatrix} \overset{L_6}{\neg Qaa} \vee \overset{L_7}{\neg Qbb} \end{bmatrix} \end{bmatrix}$$

$$\forall x^1y^1 = \begin{matrix} \forall x^1y^1 \\ \begin{bmatrix} \overset{L_1^1}{\neg Rx^1y^1} \vee \begin{bmatrix} \overset{L_2^1}{Ry^1x^1} \\ \\ \overset{L_3^1}{Qx^1y^1} \end{bmatrix} \end{bmatrix} \end{matrix}$$

$$J_2 = \begin{bmatrix} \overset{L_0}{Rab} \\ \forall xy \\ \begin{bmatrix} \overset{L_1}{\neg Rxy} \vee \begin{bmatrix} \overset{L_2}{Ryx} \\ \\ \overset{L_3}{Qxy} \end{bmatrix} \end{bmatrix} \\ \forall x^1y^1 \\ \begin{bmatrix} \overset{L_1^1}{\neg Rx^1y^1} \vee \begin{bmatrix} \overset{L_2^1}{Ry^1x^1} \\ \\ \overset{L_3^1}{Qx^1y^1} \end{bmatrix} \end{bmatrix} \\ \forall uv \\ \begin{bmatrix} \overset{L_4}{\neg Quv} \vee \overset{L_5}{Quu} \end{bmatrix} \\ \forall u^2v^2 \\ \begin{bmatrix} \overset{L_4^2}{\neg Qu^2v^2} \vee \overset{L_5^2}{Qu^2u^2} \end{bmatrix} \\ \begin{bmatrix} \overset{L_6}{\neg Qaa} \vee \overset{L_7}{\neg Qbb} \end{bmatrix} \end{bmatrix}$$

Figure 1: Vpforms for Example 1

$$J_3 = \begin{bmatrix} \begin{bmatrix} \overset{L_8}{Pa} \vee \overset{L_9}{Pb} \vee \overset{L_{10}}{Pc} \end{bmatrix} \\ \forall x \\ \begin{bmatrix} \overset{L_{11}}{\neg Px} \end{bmatrix} \end{bmatrix}$$

$$J_4 = \begin{bmatrix} \begin{bmatrix} \overset{L_8}{Pa} \vee \overset{L_9}{Pb} \vee \overset{L_{10}}{Pc} \end{bmatrix} \\ \forall x \\ \begin{bmatrix} \overset{L_{11}}{\neg Px} \end{bmatrix} \\ \forall x^1 \\ \begin{bmatrix} \overset{L_{11}^1}{\neg Px^1} \end{bmatrix} \\ \forall x^2 \\ \begin{bmatrix} \overset{L_{11}^2}{\neg Px^2} \end{bmatrix} \end{bmatrix}$$

Figure 2: Vpforms for Example 2

ample, (L_5, L_7) is a connection in J_1. For any connection $C = (L, M)$, σ_C denotes the most general unifier of the set $\{(\neg L, M)\}$. A connection (L, M) in a jform J *spans* a path P through J iff both L and M are on P.

For a set $\mathcal{M}$ of connections, $\mathcal{U}_\mathcal{M} \overset{\text{def}}{=} \{(\neg L, M) \mid (L, M) \in \mathcal{M}\}$. A *mating* $\mathcal{M}$ for a jform J is a set of connections in J such that the set $\mathcal{U}_\mathcal{M}$ is unifiable. For example, $\mathcal{M}_1 = \{(L_0, L_1), (L_3, L_4), (L_5, L_6)\}$ is a mating for J_1 and $\{(L_{11}, L_8)\}$ is a mating for J_3. For any mating $\mathcal{M}$, $\sigma_\mathcal{M}$ is the most general unifier of the set $\mathcal{U}_\mathcal{M}$.

A mating $\mathcal{M}$ for a jform J *spans* a path P through J iff there is a connection in $\mathcal{M}$ which spans P. A mating $\mathcal{M}$ for a jform J is an *extension* of a mating $\mathcal{L}$ for J iff $\mathcal{L} \subseteq \mathcal{M}$. A mating $\mathcal{M}$ for a jform J is *p-acceptable* (*path-acceptable*) iff $\mathcal{M}$ spans all the paths through J. For example, $\{(L_{11}, L_8), (L_{11}^1, L_9), (L_{11}^2, L_{10})\}$ is a p-acceptable mating for J_4. A *refutation* for a jform J is an ordered pair $(J^*, \mathcal{M})$ such that $\mathcal{M}$ is a p-acceptable mating for the amplification J^* of J. The *mating method* for automated theorem provers is a method for finding refutations for a jform J. Andrews [1981] shows that the mating method is sound and complete:

Theorem 1 Completeness and Soundness of the Mating Method *A jform J is unsatisfiable iff there is a p-acceptable mating for some amplification J^* of J.*

Path-Focused Duplication

The task of finding a refutation in the mating method can on the basis of the definition of a refutation be naturally split in two separate subtasks:

1. *Search for a p-acceptable mating for J*: enumerate all possible matings for J until either a p-acceptable mating is found or all possible matings have been considered.
2. *Replace J by some amplification of J*: duplicate some expansion node(s) in J. We consider this duplication to be a *global duplication* because it affects the entire search as described below.

This separation is the basis for the search procedures in [Andrews, 1981; Bibel, 1982] and leads to deficiencies, some of which are outlined by Stickel [1982]. Such a search process also violates a basic principle that should be followed by the search strategies; according to Palay [1980]:

If an action needs to be taken sometime in the search then do it immediately. If that action were postponed, unnecessary work may be performed. By taking the action immediately, certain information may be obtained that alters the current assumptions of the search, making explorations of other nodes unnecessary.

Path-focused duplication is an abstract procedure for finding refutations by the mating method, which incorporates amplifications (step 2 above) within the search process (step 1 above) itself. The above mentioned principle is the basis for this procedure: if a mating cannot be extended to span a path P in the existing search space (we say that the search seems to have reached an impasse), but some progress can be made by duplicating an expansion node, then duplicate immediately instead of postponing this action. We consider the path P to be the cause for this duplication. The search space grows and shrinks dynamically to respond to the changing requirements in the search: it grows to allow an impasse to be resolved and shrinks so that the amplification does not affect the entire search.

Let us look at the drawbacks of global duplication. The objective of the mating procedures is to span all the paths in the tree (jform). This can be done by independent processes—one for each path—which must maintain the compatibility of the substitutions for the shared variables. Each duplication increases the number of elements on some paths and in most cases the number of paths in the tree. For example, duplicating either of the expansion nodes $\forall xy$ or $\forall uv$ in J_1 doubles the number of paths in the resulting jform and each path has more literals. The global duplications thus seem to require greater overall effort, especially if there is a refutation in a smaller tree, which we did not find because we have not exhaustively searched all smaller trees. We address the resulting dilemma—should we or should we not go for global duplication— by *localizing* the duplication to the path that caused this duplication: it is as if each independent process amplifies its own copy of the tree, and the other processes are intentionally oblivious to these changes. We illustrate *localization* with an example:

Example 3 Consider the jform J_1 in Example 1. The mating $\mathcal{M}_1$ cannot be extended to span the path $Q =< L_0, \forall xy, L_2, L_3, \forall uv, L_5, L_7 >$. We will then duplicate one of the expansion nodes, say $\forall xy$. The node $\forall x^1 y^1$ which is a copy of $\forall xy$ is displayed in the lower left corner of Figure 1. Because we *localize* the duplication to Q, we will proceed as if the path Q has been replaced by the following two paths:

$$<L_0, \forall xy, L_2, L_3, \forall x^1y^1, L_1^1, \forall uv, L_5, L_7>$$
$$<L_0, \forall xy, L_2, L_3, \forall x^1y^1, L_2^1, L_3^1, \forall uv, L_5, L_7>$$

There are 7 other paths in J_1 that could also be extended like Q, but we will not extend them. There are 16 paths in $J_1 + \forall x^1 y^1$; the effect of localization is that the search space will have only 9 paths after duplicating the expansion node $\forall xy$.

The behavior of the search is partially determined by the number and size of the paths in the search space. Duplication causes an explosion in their number. Path-focused duplication attempts to directly control the number and size of the paths in the search space without restricting quantifier duplication; this is where the strength of the procedure lies.

We next describe *path-focused duplication*, which is a procedure for finding refutations in a jform J. We use the state-space representation for this description. A state is a pair $(Open, \mathcal{M})$ where Open is the set of partial paths not spanned by the mating and $\mathcal{M}$ is the mating that is being generated. The objective of the search is to find a sequence of operations that starts from the initial state and leads to the goal state.

1. *Initial State*:
   (a) $Open = Paths(J)$.
   (b) $\mathcal{M} = \emptyset$.
2. *Goal State*: $Open = \emptyset$.
3. *Operations*. There are two operations which transform the states:
   (a) *Extension*. Select a connection C on some $P \in Open$ such that $\mathcal{U}_{\mathcal{M} \cup \{C\}}$ is unifiable.
      i. $\mathcal{M} \longleftarrow \mathcal{M} \cup \{C\}$
      ii. $Open \longleftarrow Open - \{P\}$
   (b) *Duplication*. Select an expansion node E on some $P \in Open$. Let E_i be a copy of E. This can be a new copy or a copy that was used earlier.
   $Open \longleftarrow Open - \{P\} \cup \{P@Q \mid Q \in Paths(E_i)\}$.

We illustrate path-focused duplication with an example:

Example 4 Consider the jform J_1 in Example 1 (displayed in Figure 1). We specify a partial control strategy: use first-in-first-out (FIFO) to select elements from Open, and always make a new copy of the expansion node that is selected in the duplication operation. Figure 3 shows how path-focused duplication found a refutation for J_1. Each row in Figure 3 shows the elements in *Open* after the *operation* had been applied and also lists either the *connection* that was added to $\mathcal{M}$ or the *copy* of the expansion node that was used. If the path selected in a row was already spanned by a connection added in an earlier row, then no *operation* is shown. We would get the jform J_2 (displayed in Figure 1) if the duplications implicit in step 8 and step 10 were explicitly generated. The mating $\mathcal{M}$ consists of the connections listed in the column marked *Connection*. $(J_2, \mathcal{M})$ is a refutation for the jform J_1. There are 8 paths in J_1: $<L_0, \forall xy, L_1, \forall uv, L_4, L_6>$, $<L_0, \forall xy, L_1, \forall uv, L_4, L_7>$, $<L_0, \forall xy, L_1, \forall uv, L_5, L_6>$, $<L_0, \forall xy, L_1, \forall uv, L_5, L_7>$,
$<L_0, \forall xy, L_2, L_3, \forall uv, L_4, L_6>$,
$<L_0, \forall xy, L_2, L_3, \forall uv, L_4, L_7>$,
$<L_0, \forall xy, L_2, L_3, \forall uv, L_5, L_6>$,
$<L_0, \forall xy, L_2, L_3, \forall uv, L_5, L_7>$.
For $i \in [8]$, let P_i be the ith element in this sequence. Further, let $P_{81} = P_8 @ <\forall x^1 y^1, L_1^1>$, $P_{82} = P_8 @ < \forall x^1 y^1, L_2^1, L_3^1>$, $P_{821} = P_{82} @ <\forall u^2 v^2, L_4^2>$ and $P_{822} = P_{82} @ <\forall u^2 v^2, L_5^2>$. All the P_i's are partial paths in J_2.

	Operation	*Node*	*Connection*	*Open*
0.				$\{P_i \mid 1 \le i \le 8\}$
1.	Extension		(L_1, L_0)	$\{P_i \mid 2 \le i \le 8\}$
2.				$\{P_i \mid 3 \le i \le 8\}$
3.				$\{P_i \mid 4 \le i \le 8\}$
4.				$\{P_i \mid 5 \le i \le 8\}$
5.	Extension		(L_4, L_3)	$\{P_i \mid 6 \le i \le 8\}$
6.				$\{P_i \mid 7 \le i \le 8\}$
7.	Extension		(L_6, L_5)	$\{P_8\}$
8.	Duplication	$\forall x^1 y^1$		$\{P_{81}, P_{82}\}$
9.	Extension		(L_1^1, L_2)	$\{P_{82}\}$
10.	Duplication	$\forall u^2 v^2$		$\{P_{821}, P_{822}\}$
11.	Extension		(L_4^2, L_3^1)	$\{P_{822}\}$
12.	Extension		(L_5^2, L_7)	$\emptyset$

Figure 3: A Refutation for the Jform J_1 in Example 1

It is proved in [Issar, 1990] that path-focused duplication is a complete and sound procedure for finding refutations:

Theorem 2 *Consider any jform J.*

Soundness Theorem: *If there is a sequence of operations that starts from the initial state and leads to the goal state, then $\mathcal{M}$ is a p-acceptable mating for some amplification of J.*

Completeness Theorem: *If J is unsatisfiable, then there is a sequence of operations that starts from the initial state and leads to the goal state.*

Let us look at some advantages of path-focused duplication:

1. The number of paths considered by path-focused duplication is in most cases significantly smaller than the number of paths in the search space. For example, there are 32 paths in J_2, but path-focused duplication found a refutation by considering only 12 paths. Moreover, 10 of the 12 paths (all except P_{821} and P_{822}) were proper subsets of paths in J_2.
2. Since the effect of any duplication is local to a path, path-focused duplication can afford to be adventurous and separately try all possible duplications that might solve an impasse.
3. We can easily incorporate the set of support strategy [Wos *et al.*, 1965] into path-focused duplication: we assume that the jform is of the form $\bigwedge_{i=1}^{n} C_i$, we can identify one of the C_i's to be the *goal*, and the remaining C_i's to be the *axioms*; initialize *Open* to be the set of paths through the *goal*, and extend the *duplication* operation so that it can select the elements in the set of *axioms* also, in addition to selecting the expansion nodes. The details are provided in [Issar, 1990].
4. Path-focused duplication does not lose any advantages that might be associated with the procedures based on the level-saturation approach, since it can simulate the level-saturation procedures. The level-saturation approach specifies the copies of the expansion nodes that can be used before the search begins. If we restrict the *duplication* operation to select only these copies, then the two procedures are almost identical.

Status and Future Work

We have implemented a first-order theorem prover (without equality) based on path-focused duplication. Some of the key features in the control structure are as follows:

- Use the depth-first iterative deepening strategy to control the number of copies of an expansion node.
- Always try the extension operation before the duplication operation.
- Always make a new copy of the expansion node that is selected in the duplication operation.
- If a mating cannot be extended to a p-acceptable mating within the assigned depth, then backtrack to the last state where an alternate connection can be added.

We have tested our prover on some problems that are available in the theorem-proving literature. This program is written in CMU Common Lisp, and all experiments were performed on an IBM-RT with 12 megabytes memory. Although an earlier program for the mating method that was based on the level-saturation approach could not solve most of the benchmark problems from [Chang and Lee, 1973], our program found all of them easy. We could also easily solve the first 46 problems—most of the remaining problems are based on equality—in [Pelletier, 1986]. The table in Figure 4 presents the results for some problems: the first two problems were taken from [Andrews, 1986], the next three problems were taken from [Andrews, 1981], the sixth problem was taken from [Pelletier, 1986], and the last two problems were taken from [Stickel, 1986].

	Name	*Number of Paths*	*Number of Connections*	*Time (sec)*
1.	X2115	16	14	.11
2.	X2116	8	10	.55
3.	Thm53	8	7	.03
4.	Thm55	8	9	.05
5.	Thm56	8	7	.03
6.	Pell34	128	54	.68
7.	Has-Parts2	48	14	.35
8.	CL8	432	14	.35

Figure 4: Path-Focused Duplication

Name	*Level-saturation*		*Path-focused duplication*	
	Time (sec)	Bytes $\times 10^4$	Time (sec)	Bytes $\times 10^4$
X2115	4026.63	3×10^5	.11	.88
X2115a	5.32	17.5	.05	.39
X2116	3.32	13.0	.55	3.44
Thm53	.85	1.7	.03	.23
Thm55	3.28	8.9	.05	.35
Thm56	.82	1.5	.03	.23

Figure 5: Path-Focused Duplication vs Level-Saturation

The table in Figure 5 compares the performance of our implementation with that of an earlier program[1] which was based on the level-saturation approach. (X2115a is obtained by minimizing the scope of the quantifiers in X2115.) We use two measures for this comparison:

- *Time*. Run time for finding a refutation.
- *Space*. Number of bytes that were consed by the program as reported by Lisp.

We can get a higher-order theorem prover as in [Andrews, 1981; Pfenning, 1987], which is capable of handling equality also, by replacing the first-order unification algorithm that is used in this implementation with Huet's higher-order unification algorithm [Huet, 1975]. The implementation mentioned in this section is a prototype whose only objective is to demonstrate the effectiveness of path-focused duplication. There are many strategies described in [Issar, 1990] which can improve the performance of this program,

[1]A procedure based on the level-saturation approach is implemented in TPS [Andrews *et al.*, 1988]. We use this program for gathering data about the performance of level-saturation procedures.

but which have not been implemented yet. Some strategies, which are independent of the control structure, are as follows:

- A connection graph [Kowalski, 1975; Andrews, 1981] can be used to aid the extension operation in finding a connection on a path and the duplication operation in selecting an expansion node.
- We allow the duplication operation to select any expansion node on a path. Some of these nodes may be inappropriate because they do not have any role in the impasse that necessitated the duplication; we can restrict the duplication operation to disregard such nodes.
- The same combination of duplications may arise several times on a path. We illustrate this with an example: suppose there are three expansion nodes 1, 2, and 3 on a path; the duplication operation may try various permutations of these nodes, for example, (1, 2, 3), (1, 3, 2), and (3, 1, 2). We can restrict the duplication operation to avoid this redundancy. We have to be careful, though, because the duplication operation is not commutative, and such a restriction can affect the efficiency of the search.

There are other strategies that affect the control structure. An example of such a strategy is to use heuristic information as suggested by Stickel [1986] to perform early cutoffs within a depth and to increment the depth.

Some of the strategies in [Issar, 1990] have been implemented in a propositional theorem prover; the performance of this prover compares favorably with the performance of the resolution provers. We thus have reason to believe that these strategies will improve the performance of our first-order prover too.

Acknowledgements

First and foremost I acknowledge Professor Peter Andrews, my thesis advisor. His comments on many earlier drafts and suggestions regarding terminology helped to improve the clarity of this paper. I would also like to thank Frank Pfenning and Dan Nesmith for many discussions and for their enthusiasm about my work.

References

Andrews, Peter B.; Issar, Sunil; Nesmith, Daniel; and Pfenning, Frank 1988. The TPS theorem proving system. In *9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310. Springer-Verlag. 760–761.

Andrews, Peter B. 1980. Transforming matings into natural deduction proofs. In *5th Conference on Automated Deduction*, Lecture Notes in Computer Science 87. Springer-Verlag. 281–292.

Andrews, Peter B. 1981. Theorem proving via general matings. *Journal of ACM* 28:193–214.

Andrews, Peter B. 1986. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press.

Andrews, Peter B. 1989. On connections and higher-order logic. *Journal of Automated Reasoning* 5:257–291.

Berliner, Hans 1973. Some necessary conditions for a master chess program. In *Third International Joint Conference on Artificial Intelligence*. 77–85.

Bibel, Wolfgang; Eder, Elmar; and Fronhoefer, Bertram 1983. Towards an advanced implementation of the connection method. In *Eighth International Joint Conference on Artificial Intelligence*. 920–922.

Bibel, Wolfgang 1982. *Automated Theorem Proving.* Vieweg, Braunschweig.

Bibel, W. 1988. On the comparative complexity of resolution and the connection method. Technical Report BRC-88-6, University of British Columbia, Vancouver.

Chang, C. and Lee, R.C. 1973. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Huet, Gerard P. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science* 1:27–57.

Issar, Sunil 1990. Search strategies for general matings. Technical Report Forthcoming, Carnegie Mellon University, Pittsburgh.

Kowalski, Robert 1975. A proof procedure using connection graphs. *Journal of ACM* 22:572–595.

Loveland, Donald W. 1978. *Automated Theorem Proving: A Logical Basis.* North Holland.

Miller, Dale A. 1987. A compact representation of proofs. *Studia Logica* 46(4):347–370.

Nilsson, Nils J. 1980. *Principles of Artificial Intelligence.* Tioga Publishing Company.

Palay, Andrew J. 1980. An experimental analysis of the B* tree searching algorithm. Technical Report CMU-CS-80-106, Carnegie Mellon University, Pittsburgh.

Pelletier, Francis Jeffry 1986. Seventy five problems for testing automatic theorem provers. *Journal of Automated Reasoning* 2:191–216.

Pfenning, Frank and Nesmith, Dan 1990. Presenting intuitive deductions via symmetric simplification. In *10th International Conference on Automated Deduction.* To appear.

Pfenning, Frank 1987. *Proof Transformations in Higher-Order Logic.* Ph.D. Dissertation, Carnegie Mellon University.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *Journal of ACM* 12:23–41.

Stickel, Mark E. 1982. A nonclausal connection-graph resolution theorem-proving program. In *AAAI-82 National Conference on Artificial Intelligence*. 229–233.

Stickel, Mark E. 1986. A PROLOG technology theorem prover: implementation by an extended PROLOG compiler. In *8th International Conference on Automated Deduction*, Lecture Notes in Computer Science 230. Springer-Verlag. 573–587.

Wallen, Lincoln A. 1990. *Automated Proof Search in Non-Classical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics.* The MIT Press.

Wos, Lawrence; Robinson, George A.; and Carson, Daniel F. 1965. Efficiency and completeness of the set of support strategy in theorem proving. *Journal of ACM* 12:536–541.