Mechanization of Analytic Reasoning about Sets

Alan F. McMichael

AT&T Bell Laboratories
480 Red Hill Rd., 1A-214, Middletown, NJ 07748

Abstract

Resolution reasoners, when applied to set theory problems, typically suffer from "lack of focus." MARS is a program that attempts to rectify this difficulty by exploiting the definition-like character of the set theory axioms. As in the case of its predecessor, SLIM, it employs a tableau proof procedure based on binary resolution, but MARS is enhanced by an equality substitution rule and a device for introducing previously proved theorems as lemmas. MARS's performance compares favorably with that of other existing automated reasoners for this domain. MARS finds proofs for many basic facts about functions, construed as sets of ordered pairs. MARS is being used to attack the homomorphism test problem, the theorem that the composition of two group homomorphisms is a group homomorphism.

Introduction

Set theory is a notoriously difficult domain for automated reasoning programs. The basic axioms of set theory are many and complex, yet they can be formulated quite naturally in terms of three primitive predicates: "is a set, "is a member of," and "is equal to." (These are my preferred primitives; other choices are possible. My choice corresponds to a "Goedelian" set theory (Wos 1987).) With so many axioms and so few primitives, numerous deduction possibilities are always available. To make matters worse, the set theory axioms support the definition of numerous derivative concepts, such as *ordered pair*, *function*, and *cardinal number* — indeed, they suffice for the definition of all the concepts of ordinary mathematics. So an automated reasoner is faced at once with both many deduction possibilities, engendered by complex axioms and a small basic vocabulary, and the problem of reasoning with a large derivative vocabulary.

The difficulty of set theory extends to even the simplest problems in that domain. One notable early attempt to prove the commutativity of set intersection using hyperresolution resulted in failure (McCharen, Overbeek, & Wos 1976). With a program based on linear resolution and a simple weighting strategy — perhaps an approach more suitable to this kind of problem — I have been able to find an automatic proof after about 130 inferences. But on a problem of the next level of difficulty, namely, the associativity of intersection, the same program required 19,000 inferences (McMichael 1990). Obviously, such a program is confined to trivial results and cannot be adapted to problems of modest difficulty.

Programs that have done better — and there are some (e.g., Bledsoe 1977, Brown 1978 & 1986, and Pastre 1978 & 1989) — have typically diverged sharply from the resolution framework, achieving greater power at the expense of added complexity in the deductive mechanism. Evaluation of the success of such programs involves some subtleties. Progress in theorems proved must be discounted by added complexity, for there is always suspicion that the progress has come by *ad hoc* means. Also, complex deduction mechanisms tend to resist formal analysis.

This paper describes a program, called MARS (Mechanization of Analytic Reasoning about Sets), that might be said to lie midway between the resolution and nonresolution approaches. It employs a tableau-style proof procedure based on binary resolution and a restricted form of equality substitution. It is designed to attack only a fragment of set theory problems, those I shall designate as "analytic." MARS's deduction scheme is more complicated than most resolution schemes — in this respect it resembles the nonresolution approaches — but perhaps not so complicated as to resist evaluation and formal analysis.

In respect of its ability to solve set theory test problems, MARS compares favorably with existing resolution reasoners and with suitably simple and systematic nonresolution reasoners. For example, the ordered pairs theorem used by F. Brown to demonstrate the power of his own reasoning program (Brown 1986) is solved by MARS. MARS goes beyond this theorem to prove basic theorems about functions — where functions are understood to be sets of ordered pairs — including a variant of the theorem that the composition of two homomorphisms is a homomorphism. This last theorem is a well-known test problem (Wos 1987). It is not a *really* hard problem for humans, since it is a prime example of an "analytic" problem. Yet its solution — or, more precisely, its near solution, since MARS cannot yet handle Wos's version — is significant from the standpoint of automation.

The first portion of this paper reviews MARS's tableau proof procedure, a procedure first employed in MARS's predecessor, SLIM ("Simple Logical Inference Machine," McMichael 1990). This is followed by a discussion of MARS enhancements, namely a rule for equality and a scheme for handling lemmas (previously proved theorems), and of their effectiveness on test problems. Special attention is given to the homomorphism theorem.

The Tableau Proof Mechanism

Consider the nine set theory clauses relevant to the problem of proving the commutativity of set intersection:

$x = y$ if and only if $x \subseteq y$ and $y \subseteq x$

=1. $x = y \vee x \not\subseteq y \vee y \not\subseteq x$
=2. $x \neq y \vee x \subseteq y$
=3. $x \neq y \vee y \subseteq x$

$x \subseteq y$ if and only if for all z, if $z \in x$, then $z \in y$

⊆1. $x \subseteq y \vee ss(x,y) \in x$
⊆2. $x \subseteq y \vee ss(x,y) \notin y$
⊆3. $x \not\subseteq y \vee z \notin x \vee z \in y$

$x \in y \cap z$ if and only if $x \in y$ and $x \in z$

∩1. $x \in y \cap z \vee x \notin y \vee x \notin z$
∩2. $x \notin y \cap z \vee x \in y$
∩3. $x \notin y \cap z \vee x \in z$

For a moderately trained human reasoner, a proof of commutativity of intersection from these clauses is routine. The clauses, in the indicated groups of three, function as "definitions" of the concepts of set equality, subset, and intersection, and the proof is constructed merely by negating the desired conclusion, expanding according to the definitions, and making obvious inferences of propositional logic. By an *analytic* problem, I mean one that can be solved by a proof of this sort, that is, by a proof constructed solely by means of definition expansion and involving at worst a multiplicity of possible free variable bindings to be tried (a complication not present in this first example).

My first experiments with commutativity involved linear resolution. Linear resolution is unable to emulate the idea of definition expansion and consequently suffers from *lack of focus*, a well-known obstacle for reasoning programs (Wos 1987). From "$a \cap b \neq b \cap a$", my linear resolution program deduces "$a \cap b \not\subseteq b \cap a \vee b \cap a \not\subseteq a \cap b$" and then proceeds to eliminate the first "$\not\subseteq$" and conclude "$ss(a \cap b, b \cap a) \notin a \cap b \vee b \cap a \not\subseteq a \cap b$". But now that "$\not\subseteq$" has been lost from the left branch of the proof tree, linear resolution succeeds in returning to the other half of the definition of "$\not\subseteq$" (⊆2) only after a blind search through the nonleading literals in this clause set.

Using binary resolution and tableau proof, this inadequacy can be rectified. In a tableau proof, any literal on the currently selected open proof branch is in principle available to extend the branch. Thus, when the current branch is extended via clause ⊆1, the next inference need not be made, as in linear resolution, using the result of ⊆1's application but instead may be an extension via clause ⊆2 and that literal's parent. Indeed, such a choice corresponds to the strategy of definition expansion.

Effective implementation of this proof concept involves three guiding ideas:

1. Tableau Idea — The notion of an available literal is expanded to include all literals on the current branch of the proof tree.
2. Definition Expansion Idea — Clauses that serve as definitions should be used, whenever possible, to expand the defined notions (which, by convention, are displayed in the *leading* literals of these clauses).
3. Content Exhaustion Idea — Once a literal has been expanded fully according to the clauses constituting a definition, no further inferences involving it should be tried on the current branch.

A binary resolution tableau guided by these ideas leads directly to a solution of the commutativity problem. Figure 1 shows the complete tableau. When the symbol 'ss(")' occurs in a branch, it abbreviates the more lengthy skolem term introduced above it. The X's symbolize contradictions.

The same guiding ideas can be used to solve many and more difficult analytic problems. However, it is ultimately necessary to supplement them with several refinements:

1. Repetitions — An extension that results in a literal identical to one of its branch predecessors is *retracted.* A redundant literal produced indirectly by variable binding is *blocked* from further expansion.
2. Irrelevancies — The root of the proof tree consists of the clauses resulting from the negation of the theorem (with minor exceptions). It constitutes the "set-of-support" of the problem, as opposed to the "auxiliary clauses" provided by the set theory axioms. However, a literal may be generated later in the tableau, such as "$a \subseteq a$", which is in fact a logical consequence of auxiliary clauses. As in the case of repetitious literals, these may be retracted or blocked. (This is an extension of the set-of-support

1. $a \cap b \neq b \cap a$
(negated theorem)

2. $a \cap b \not\subseteq b \cap a$ (1,=1)			v	3. $b \cap a \not\subseteq a \cap b$		
4. $ss(a \cap b, b \cap a) \in a \cap b$ $(2,\subseteq 1)$				11. $ss(b \cap a, a \cap b) \in b \cap a$ $(3,\subseteq 1)$		
5. $ss(a \cap b, b \cap a) \notin b \cap a$ $(2,\subseteq 2)$				12. $ss(b \cap a, a \cap b) \notin a \cap b$ $(3,\subseteq 2)$		
6. $ss(") \notin b$ $(5,\cap 1)$	v	7. $ss(") \notin a$		13. $ss(") \notin a$ $(12,\cap 1)$	v	14. $ss(") \notin b$
8. $ss(") \in a$ $(4,\cap 2)$		10. $ss(") \in a$ $(4,\cap 2)$		15. $ss(") \in b$ $(11,\cap 2)$		17. $ss(") \in b$ $(11,\cap 2)$
9. $ss(") \in b$ $(4,\cap 3)$		XXX (10,7 ancestor rsl.)		16. $ss(") \in a$ $(11,\cap 3)$		XXX (17,14 ancestor rsl.)
XXX (9,6 ancestor rsl.)				XXX (16,13 ancestor rsl.)		

Figure 1: Proof of Commutativity Theorem

idea (Wos et al. 1967), so may be regarded as an answer to "Research Problem 1" (Wos 1987).)

3. Skipping Unused Branches — Extension of the proof tree by means of an auxiliary clause containing three or more literals results in new branches. If one of the new branches is ultimately closed without using the literal at the beginning of the branch, then the other branches may be skipped.

4. Free Variables: Controlled Introduction, Binding, and Backtracking — The tableau proof method is complete even if variables are required to bind only with closed terms occurring previously in the branches in which the variables appear. Completeness is lost if inferences are confined to definition expansions. However, for analytic problems, the suggested restriction on variable bindings makes sense and can be used to guide proofs. Accordingly, clauses that result in the production of new free variables are introduced one at a time; the free variables of one are required to be bound before another is tried. Backtracking may occur before the correct clause is found and suitable bindings made.

The three guiding ideas and four refinements were first implemented by MARS's predecessor, SLIM. SLIM is able to prove many simple set theory facts, as Figure 2 shows. Equality substitution is needed for the last problem in the table. SLIM has no special rules for equality, so is compelled to solve the problem using equality substitution axioms. The specific clauses required pertain to the membership primitive:

$$x \notin y \lor z \in y \lor x \neq z$$
$$x \in y \lor z \notin y \lor x \neq z$$

Substitution axioms are known to be a clumsy means for handling equality. Since equality figures prominently in the next set of problems on my agenda, namely, basic theorems about functions, SLIM has clearly reached the limits of its competence.

Another shortcoming of SLIM is its inability to make use of *lemmas* — previously proved theorems — as shortcuts in the construction of new proofs. One way this crops up is in SLIM's check for irrelevant literals: SLIM treats the literal as a potential theorem and actually attempts a little proof of it from the auxiliary axioms. Successful proof means the literal is irrelevant. But a necessary result of this procedure is that many inferences are expended merely to check, for each new literal, whether it is one of a handful of one-literal theorems. If instead one-literal theorems were stored and used as lemmas on demand, all these inferences would be avoided. Moreover, these lemmas could be used also to close branches in the main proof tree.

Equality and Lemmas in MARS

MARS is an enhancement of SLIM that incorporates special rules for equality and lemmas. In order to preserve the desirable features of SLIM, the design of these rules has been approached conservatively.

SLIM			
Theorem	Inferences In Solution	Total Inferences	Axiom Set Notes
$P(a) \cap P(b) = P(a \cap b)$	39	419	No "set" predicate
$\{a,b\} = \{a,c\} \rightarrow b = c$	106	922	(Naive Set Theory).
$b = c \rightarrow \{a,b\} = \{a,c\}$	118	1357	Definition axioms for =, but
$<a,b> = <c,d> \rightarrow a = c$ & $b = d$	555	23573	no substitution axioms.
$a = c$ & $b = d$ & set(a) & set(b) & set(c) & set(d) $\rightarrow <a,b> = <c,d>$	687	25893	Set predicate added.
ordered-pair(a) & $a = <b,c> \rightarrow b \subseteq$ first(a)	151	1536	
ordered-pair(a) & $a = <b,c> \rightarrow c \subseteq$ second(a)	645	32006	
set(m) & set(n) $\rightarrow m \times n \subseteq P(P(m \cup n))$	233	15331	Substitution axioms added.

Figure 2: SLIM's Performance

The MARS substitution rule for equality is applied to ground literals only (literals with no free variables). Moreover, it is applied only when the terms involved in an equality literal are *not* reducible according to the set theory axioms. (That is, their main function symbols, unlike P, $\cap$, $\cup$, or {_}, are not "defined" by the axioms.) The rule states that the *lesser* of the two terms in the equality, according to a natural complexity ordering, may replace the greater everywhere it occurs in another literal. If the equality literal is at the tip of a proof branch, then the substitution rule is applied to all of its predecessors on the branch, producing a group of successor literals under the equality and effectively removing the greater term from further consideration in the branch. If instead the equality literal is higher up in the branch, then the substitution is made into the tip goal alone to produce a single successor. If two equalities are involved, the one higher up in the branch takes precedence. Figure 3 illustrates the substitution rule. The MARS equality rule shares features of ordered paramodulation, simultaneous paramodulation, and demodulation (Bachmair and Granzinger 1990, Benanav 1990, and Wos et al. 1967) but is simpler and more restricted.

The equality rule enables MARS, even without use of lemmas, to solve problems beyond SLIM's reach. For example, with functions defined as sets of ordered pairs, MARS is able to prove the theorem: func(f) & $<a,b> \in f \rightarrow b = f(a)$, where "f(a)" is an abbreviation for explicit reference to the result of function application, "result(f,a)", that is actually required. (Wos 1987 uses "image(a,f)" to express this notion.) MARS also solves, without lemmas, all the problems SLIM solves.

A naive strategy for employing lemmas in proofs can drastically reduce an automated reasoner's effectiveness. This happens if the application of a lemma turns out not to be conclusive and essentially the same line of reasoning is reproduced later from another lemma or from the basic axioms. Repeated instances of such failure result in exponentially bad performance. This difficulty can be ameliorated, if not evaded altogether, by concentrating on *demonstrably successful* lemma applications. A paradigm case of this is *the use of a single-literal lemma to close a branch without binding any variables in the proof tree.* Such an inference creates no backtracking point. Since no variable binding occurs, there is no need to try an alternative means of closing the branch.

In the case of *multi-literal* lemmas that bind no proof variables, there is a danger of producing unnecessary extensions and splits in the proof tree. In some cases, the danger is minimal. Consider the clause:

$$x \in \{x,y\} \vee \neg set(x) \vee \neg set(y)$$

The first literal "narrowly misses" being a theorem of set theory ("Goedelian" set theory). The supplementary facts needed are that x and y are sets not "proper classes" (intuitively, collections "too large" to constitute sets). But since proper classes rarely figure in elementary theorems and such supplementary facts are often available, MARS can make good use of lemmas of this sort. Let us lump these literals together with the true single-literal lemmas and call them *unitary lemmas.*

My first experiments with lemmas in MARS involved only unitary lemmas whose proofs are shallow. These are exactly the sorts of lemmas that are detected by SLIM's irrelevancy check. In fact, my first list was compiled from literals proved irrelevant by SLIM and MARS. For a practical solution to the homomorphism test problem, however, it proves necessary to give MARS one multi-literal lemma (in four clauses) which is not unitary and which does not have a shallow proof. This lemma is none

Substitution FROM Tip	Substitution INTO Tip
1. $\{d,b\} \subseteq \{d,c\}$	1. $ss10 = b$
2. $\{d,c\} \subseteq \{d,b\}$	2. $ss10 \notin \{b,c\}$
3. $d \in \{d,b\}$	3. $b \notin \{b,c\}$ (1,2)
4. $c \in \{d,b\}$	
5. $c = d$ v $c=b$	
6. $c \in \{c,b\}$ (5,3)	
7. $\{c,c\} \subseteq \{c,b\}$ (5,2)	
8. $\{c,b\} \subseteq \{c,c\}$ (5,1)	

Figure 3: Substitution Rule Examples

other than the fundamental property of ordered pairs which was proved by F. Brown's program (Brown 1986) and which served as a test problem for SLIM:

> If $\langle x,y \rangle = \langle u,v \rangle$, then $x = u$ and $y = v$ or (degenerate case) one of x and y is not a set and one of u and v is not a set.

$\langle x,y \rangle \neq \langle u,v \rangle$ v –set(x) v –set(y) v $x = u$
$\langle x,y \rangle \neq \langle u,v \rangle$ v –set(x) v –set(y) v $y = v$
$\langle x,y \rangle \neq \langle u,v \rangle$ v –set(u) v –set(v) v $x = u$
$\langle x,y \rangle \neq \langle u,v \rangle$ v –set(u) v –set(v) v $y = v$

Figure 4 gives test problem data for MARS, culminating in the homomorphism theorem. MARS does not quite solve the homomorphism problem originally posed by Wos. Instead, it proves that a composition of triadic relation homomorphisms is a homomorphism. A group is a binary operation over a given domain, and thus is a special case of a triadic relation. Indeed, a triadic relation homomorphism between groups will also be a group homomorphism. MARS, however, cannot prove this. It can show that the triadic relation homomorphism has one of the crucial properties of a group homomorphism — that is the content of the last problem in the table. But to prove all the relevant properties, MARS would have to be able to reason effectively about the group closure property, and it currently does not. Thus, in the last problem, an instance of closure is assumed in the hypothesis of the theorem. There is reason to hope, however, that this difficulty will be soon overcome.

The experience with MARS indicates that the use of lemmas is an unavoidable device. If this is indeed so, then we are forced to confront questions about what constitutes "fair play" in automated theorem-proving. At a minimum, of course, the program that proves the theorem should also be able to prove the lemmas it invokes, and the invocation of lemmas should be an automatic process. MARS satisfies these minimum requirements. On the other hand, it would be nice to have a program that automatically selects lemmas as it gains experience with various deduction problems. MARS is not such a program; it is not like "AM" (Davis & Lenat 1982). MARS could easily be modified to prove and collect the shallow unitary lemmas, but it is not clear how to automate the selection of deeper, multi-literal lemmas. Nevertheless, I think the *actual* selection of lemmas for the homomorphism problem is sufficiently conservative and does not undermine the significance of MARS's success.

Undoubtably the greatest shortcoming of MARS is its inability to solve synthetic problems. Not all set theory problems can be solved by definition expansion coupled with, whenever free variables appear in the proof tree, trial of alternative inferences that bind those variables. My favorite example is Cantor's theorem, which states that there can be no function on a set S whose range includes the whole power set of S. The proof proceeds by assuming that there is such a function and showing that the existence of that function implies the existence of a subset of S that cannot possibly be in the range of the function — a contradiction. The step from the existence of the function to the existence of the recalcitrant subset is one MARS cannot reproduce.

There are two MARS inference restrictions that stand in the way of synthetic deductions, namely, (1) the use of "definition" axioms in one direction only, and (2) the requirement of controlled binding of free variables. Since (1) is MARS's prime device for attacking the problem of *focus*, my plan is to revise (2) instead. But certainly I have not gone far toward implementing this plan. Much more experimental and theoretical study needs to be done.

MARS			
Theorem	Inferences In Solution	Total Inferences	Axiom Set and Problem () Notes
func(f) & <a,b> ∈ f → b = f(a)	318	4148	Set predicate present. No = substitution axioms (unnecessary). No lemmas.
func(f) & <a,b> ∈ f → b = f(a)	155	825	Unitary lemmas.
func(f) & func(g) → func(f∘g)	5884	49310	
func(f) & <a,b> ∈ f → b = f(a)	20	75	Ordered pair lemma added.
func(f) & func(g) → func(f∘g)	97	439	
func(f) & func(g) & one-to-one(f) & one-to-one(g) → one-to-one(f∘g)	190	796	
homomorphism(f,p,q) & homomorphism(g,q,r) → homomorphism(f∘g,p,r)	1925	40765	(Triadic relation homomorphism, more general than Wos's group homomorphism.)
homomorphism(f,p,q) & func(p) & func(q) & a ∈ domain(f) & b ∈ domain(f) & p(a,b) ∈ domain(f) & <f(a),f(b)> ∈ domain(q) → q(f(a),f(b)) = f(p(a,b))	41	263	(Triadic relation homomorphism has group homomorphism property.)

Figure 4: MARS's Performance

Appendix: Homomorphism Problem Clauses

set equality
x ≠ y v sub(x,y)
 & x ≠ y v sub(y,x)
x = y v -sub(x,y) v -sub(y,x)

subset
-sub(x,y) v z ∉ x v z ∈ y
sub(x,y) v ss(x,y) ∈ x
 & sub(x,y) v ss(x,y) ∉ y

pair set
x ∉ pr(y,z) v set(y)
 & x ∉ pr(y,z) v set(z)
 & x ∉ pr(y,z) v x = y v x = z
x ∈ pr(y,z) v x ≠ y v -set(y) v -set(z)
 & x ∈ pr(y,z) v x ≠ z v -set(y) v -set(z)
set(pr(x,y))

singleton
x ∉ sing(y) v set(y)
 & x ∉ sing(y) v x = y
x ∈ sing(y) v x ≠ y v -set(y)
set(sing(x))

ordered pair
x ∉ <y,z> v x ∈ pr(sing(y),pr(y,z))
x ∈ <y,z> v x ∉ pr(sing(y),pr(y,z))
set(<y,z>)

is an ordered pair, 1st(sx), 2nd(sy)
-isop(x) v x = <sx(x),sy(x)>
isop(x) v x ≠ <y,z> v -set(y) v -set(z)
set(sx(x))
set(sy(x))

relation
-rel(x) v y ∉ x v isop(y)
rel(x) v sr(x) ∈ x
 & rel(x) v -isop(sr(x))
set(sr(x))

function and value(sv)

```
-func(x) v rel(x)
 & -func(x) v <y,z> ∉ x v z = sv(x,y)
func(x) v -rel(x) v <sf(x),sA(x)> ∈ x
 & func(x) v -rel(x) v <sf(x),sv(x,sf(x))> ∈ x
 & func(x) v -rel(x) v sA(x) ≠ sv(x,sf(x))
set(sv(x,y))
set(sf(x))
set(sA(x))
```

composition of functions

```
x ∉ cmp(u,v) v isop(x)
 & x ∉ cmp(u,v) v <sx(x),sv(u,sx(x))> ∈ u
 & x ∉ cmp(u,v) v <sv(u,sx(x)),sy(x)> ∈ v
x ∈ cmp(u,v)) v -isop(x) v
   <sx(x),sv(u,sx(x))> ∉ u v
   <sv(u,sx(x)),sy(x)> ∉ v)
set(cmp(u,v))
```

is an ordered triple

```
-istrip(x) v x = <<sx(sx(x)),sy(sx(x))>,sy(x)>
istrip(x) v x ≠ <u,v> v -isop(u) v -set(v)
```

is a triadic relation

```
-rel3(x) v y ∉ x v istrip(y)
rel3(x) v s3(x) ∈ x
 & rel3(x) v -istrip(s3(x))
set(s3(x))
```

homomorphism

```
set(sE(x,w1,w2))
set(sF(x,w1,w2))
set(sG(x,w1,w2))
-homo(x,w1,w2) v func(x)
 & -homo(x,w1,w2) v rel3(w1)
 & -homo(x,w1,w2) v rel3(w2)
 & -homo(x,w1,w2) v
   <<u1,u2>,u3> ∉ w1 v
   <u1,sv(x,u1)> ∉ x v
   <u2,sv(x,u2)> ∉ x v
   <u3,sv(x,u3)> ∉ x v
   <<sv(x,u1),sv(x,u2)>,sv(x,u3)> ∈ w2
homo(x,w1,w2) v -func(x) v -rel3(w1) v -rel3(w2) v
   <<sE(x,w1,w2),sF(x,w1,w2)>,sG(x,w1,w2)> ∈ w1
 & homo(x,w1,w2) v -func(x) v -rel3(w1) v -rel3(w2) v
   <sE(x,w1,w2),sv(x,sE(x,w1,w2))> ∈ x
 & homo(x,w1,w2) v -func(x) v -rel3(w1) v -rel3(w2) v
   <sF(x,w1,w2),sv(x,sF(x,w1,w2))> ∈ x
 & homo(x,w1,w2) v -func(x) v -rel3(w1) v -rel3(w2) v
   <sG(x,w1,w2),sv(x,sG(x,w1,w2))> ∈ x
 & homo(x,w1,w2) v -func(x) v -rel3(w1) v -rel3(w2) v
   <<sv(x,sE(x,w1,w2)),sv(x,sF(x,w1,w2))>,
      sv(x,sG(x,w1,w2))> ∉ w2
```

References

Bacmair, L., and Granzinger, H. 1990. On Restrictions of Ordered Paramodulation with Simplification. In Proceedings of the Tenth International Conference on Automated Deduction, 427-441. Springer-Verlag.

Benanav, D. 1990. Simultaneous Paramodulation. In Proceedings of the Tenth International Conference on Automated Deduction, 442-455. Springer-Verlag.

Bledsoe, W. W. 1977. Non-resolution Theorem Proving. *Artificial Intelligence Journal* 9:1-35.

Brown, F. M. 1978. Towards the Automation of Set Theory and Its Logic. *Artificial Intelligence Journal* 10:281-316.

Brown, F. M. 1986. An Experimental Logic Based on the Fundamental Deduction Principle. *Artificial Intelligence Journal* 30:117-263.

Davis, R., and Lenat, D. 1982. *Knowledge-Based Systems in Artificial Intelligence*. McGraw Hill.

McCharen, J., Overbeek, R., and Wos, L. 1976. Problems and Experiments for and with Automated Theorem-Proving Programs. *IEEE Transactions on Computers* C-25:773-782.

McMichael, A. 1990. SLIM: An Automated Reasoner for Equivalences, Applied to Set Theory. In Proceedings of the Tenth International Conference on Automated Deduction, 308-321. Springer-Verlag.

Pastre, D. 1978. Automated Theorem Proving in Set Theory. *Artificial Intelligence Journal* 10:1-27.

Pastre, D. 1989. MUSCADET: An Automated Theorem Proving System using Knowledge and Metaknowledge in Mathematics. *Artificial Intelligence Journal* 38:257-318.

Wos, L., Robinson, G., and Carson, D. 1965. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *Journal of the Association for Computing Machinery* 12:536-541.

Wos, L., Robinson, G., Carson, D., and Shalla, L. 1967. The Concept of Demodulation in Theorem Proving. *Journal of the Association for Computing Machinery* 14:698-709.

Wos, L. 1987. *Automated Reasoning: Thirty Three Basic Research Problems*. Prentice-Hall.