The Automatic Acquisition of Proof Methods*

Kurt Ammon
Fibigerstr. 163, D-2000 Hamburg 62
Federal Republic of Germany

Abstract

The SHUNYATA program constructs proof methods by analyzing proofs of simple theorems in mathematical theories such as group theory and uses these methods to form proofs of new theorems in the same or in other theories. Such methods are capable of generating proofs of theorems whose complexity represents the state of the art in automated theorem proving. They are composed of elementary functions such as the union of sets and the subset relation. Elementary knowledge about these functions such as descriptions of their domains and their ranges forms the basis of the method acquisition processes. These processes are controlled genetically, which means that SHUNYATA, starting from scratch, constructs a sequence of more and more powerful partial methods each of which forms the basis for the construction of its successor until a complete method is generated.

1 Introduction

Mathematicians are often capable of generating many proofs in a mathematical theory such as group theory on the basis of a few methods which can be arrived at by analyzing a few simple proofs. This phenomenon formed a starting point for the development of the SHUNYATA program which automatically acquires proof methods by analyzing proofs and then uses these methods to generate proofs of new theorems. An investigation of the processes in this program provides an insight into method acquisition and proof discovery processes in the complex domain of higher mathematics. This paper focuses on the method acquisition processes in SHUNYATA: Sections 2 and 3 give an example of a proof and a proof method. Section 4 describes a method acquisition procedure which constructs the proof method from the proof. Section 5 shows how the method can be used to generate proofs of new theorems. Section 6 discusses the generality and the power of the procedure. Finally, Section 7 compares this work with related research and Section 8 summarizes the most important results.

2 Proof

The input of the learning process consists of a set of axioms, a theorem, and its proof. This section gives an example of an input. The axiomatization of group theory chosen here defines groups as precisely those algebraic structures for which solutions to linear equations are guaranteed, rather than the usual axiomatization for which linear equation solvability is a theorem. The author began his investigations with this axiomatization, but it has no special relevance.

A group is a set with three binary functions f, g, and h. We write xy for $f(x,y)$, where x and y are terms. The axioms are:

1. $(xy)z = x(yz)$
2. $g(x,y)x = y$
3. $xh(x,y) = y$

An example of a group is the set of integers with addition where the second and the third axiom mean that the equations $sx = y$ and $xs = y$ have solutions s. A theorem is: For all elements a and b of a group, the equation

$$g(a,a)b = b$$

holds. It implies that there is a left identity in a group (see [MacLane & Birkhoff 67, pp. 78-79, Exercise 8]). In order to improve the readability of this paper, we regard $x(yz) = (xy)z$, $y = g(x,y)x$, and $y = xh(x,y)$ as additional axioms.

A proof for the theorem $g(a,a)b = b$ in ordinary mathematical representation is: Because of axiom 3, the equation $g(a,a)b = g(a,a)(ah(a,b))$ holds. Because of axiom 1, the equation $g(a,a)(ah(a,b)) = (g(a,a)a)h(a,b)$ holds. Because of axiom 2, the equation $(g(a,a)a)h(a,b) = ah(a,b)$ holds. Because of axiom 3, the equation $ah(a,b) = b$ holds. Because of the transivity of the equality relation, the above equations imply the equation $g(a,a)b = b$ which was to be proved. A computational representation of this proof is given in Table 1. Each row in the table is a proof step, i.e., the proof consists of four steps. A *proof step* is a tuple whose members are an equation, a term, a pointer to a subterm of this term, an axiom, and a substitution for the variables of this axiom. A *pointer* is a natural number or a list of natural numbers that points to a subterm of a term. The empty list points to a term itself. The equations in the first column are the equations in the proof in ordinary representation. The first equation $g(a,a)b = g(a,a)(ah(a,b))$ in the proof is generated as follows: Its left side is equal to the term $g(a,a)b$ in the second column. The application of the substitution $\{a/x,\ b/y\}$ for the variables x and y to the third axiom yields the equation $ah(a,b) = b$. The replacement of the subterm b in the term $g(a,a)b$ the pointer 2 points to by the term $ah(a,b)$ yields the term $g(a,a)(ah(a,b))$ which is equal to the right side of the first equation in the proof. The other equations in the proof are generated analogously.

3 Proof Method

The output of the learning process is a proof method which generates a proof from a set of axioms and a theorem in the input of the learning process. This section gives an example of a method. The evaluation of proof methods, i.e., the performance component of SHUNYATA, is described in the appendix. Section 4 gives a method acquisition procedure which generates the method from the proof in Table 2. Section 5 describes how the method can be used to form proofs of new theorems.

The method consists of two operators which generate finite sets of proof steps. It is given in Table 2. The second operator has priority, which means that the first operator is applied only if the second one generates no proof steps. The proof steps generated by these operators are appended to a list which is empty at the beginning. It is called *partial proof*. The operators can

*This work, in whole or in part, describes components of machines or processes protected by one or more patents or patent applications in Europe, Japan, the United States of America, or elsewhere. Infomation is available from the author.

Equation	Term	Pointer	Axiom	Substitution
$g(a,a)b = g(a,a)(ah(a,b))$	$g(a,a)b$	2	3	$\{a/x,\ b/y\}$
$g(a,a)(ah(a,b)) = (g(a,a)a)h(a,b)$	$g(a,a)(ah(a,b))$	()	1	$\{g(a,a)/x,\ a/y,\ h(a,b)/z\}$
$(g(a,a)a)h(a,b) = ah(a,b)$	$(g(a,a)a)h(a,b)$	1	2	$\{a/x,\ a/y\}$
$ah(a,b) = b$	$ah(a,b)$	()	3	$\{a/x,\ b/y\}$

Table 1: A proof for the theorem $g(a,a)b = b$

$\{(e,t,p,a,s) \mid$	*is-a-proof-step* $(e,t,p,a,s) \wedge$ *left-side* $(e) =$ *left-side* $(THEOREM) \wedge$ $a \in \{AXIOM\text{-}1,\ AXIOM\text{-}2,\ AXIOM\text{-}3\} \wedge$ *substituents* $(s) \subseteq$ *variables* $(THEOREM)\}$
$\{(e,t,p,a,s) \mid$	*is-a-proof-step* $(e,t,p,a,s) \wedge$ *left-side* $(e) \in$ *right-sides* (*equations* $(PARTIAL\text{-}PROOF)) \wedge$ $a \in \{AXIOM\text{-}1,\ AXIOM\text{-}2,\ AXIOM\text{-}3\} \wedge$ *variables* (*left-side* $(a)) \supseteq$ *variables* (*right-side* $(a)) \wedge$ *right-side* $(e) \notin$ *left-sides* (*equations* $(PARTIAL\text{-}PROOF))\}$

Table 2: A proof method consisting of two set operators

be regarded as rules of inference in the sense that the equations in the proof steps they produce are valid. The first operator generates the set of all proof steps (e,t,p,a,s), where e is an equation, t is a term, p is a pointer to subterm of this term, a is an axiom, and s is a substitution for the variables in this axiom, that satisfy the following constraints: The left side of the equation e is equal to the left side of the theorem, the axiom a is the first, second, or third axiom, and the substituents in the substitution s are variables contained in the theorem. Roughly speaking, the first operator generates equations whose left side is equal to the left side of the theorem and whose right side is constructed on the basis of substitutions whose substituents are variables contained in the theorem. The first proof step of the proof in Table 1 satisfies the constraints in the first operator: The left side $g(a,a)b$ of the equation in this proof step is equal to the left side of the theorem $g(a,a)b = b$, the axiom is the third axiom, and the substituents a and b are variables contained in the theorem $g(a,a)b = b$. The second operator in Table 2 generates the set of all proof steps (e,t,p,a,s) that satisfy the following constraints: The left side of the equation e is equal to the right side of an equation in the partial proof, the axiom a is the first, second, or third axiom, the variables in the right side of the axiom a are contained in its left side, and the right side of the equation e is different from the left sides of the equations in the partial proof. Roughly speaking, the second operator generates equations whose left side is the right side of an equation in the partial proof and whose right side is not longer than its left side. The second, the third, and the fourth proof step of the proof in Table 1 satisfy the constraints in the second operator: For example, the left side $(g(a,a)a)h(a,b)$ of the equation in the third proof step is equal to the right side of the preceding equation, the axiom is the second axiom, the variable y in the right side of the axiom is contained in its left side, and the right side $ah(a,b)$ of the equation is different from the left sides of the preceding equations. The evaluation of the two operators, i.e., the performance component of SHUNYATA, is described in the appendix. They yield a set of proof steps containing the proof steps in Table 1.

The application of the method to the theorem $g(a,a)b = b$ produces eighteen proof steps whose equations are given in Table 3. The equations required for the proof are underlined. The equations in the first column are generated by the first operator and the equations in the other columns by the second operator of the proof method. The equations in the first column are produced by the second and the third axiom. The equations in the second column are produced by the first axiom. The equation in the third column is produced by the second axiom and the equation in the fourth column by the third axiom.

4 Method Acquisition Procedure

This section describes a method acquisition procedure which constructs the method in Table 2 from the proof in Table 1, i.e., it describes the learning procedure of SHUNYATA. This procedure analyzes the proof steps one by one and, starting from scratch, constructs partial methods which generate the proof steps in the proof up to the *current proof step*, i.e., the one that is presently being analyzed. The partial methods are lists of set operators such as the operators in Table 2. The analysis of a single proof step is performed as follows: First, the procedure applies the partial method within a given time interval. If this yields the current proof step, the next proof step is processed. Otherwise, an elementary analysis of the current proof step is performed which yields a set operator generating this proof step. In a simplification process, the unessential constraints of the set operator are deleted. Then, the procedure attempts to unify the resulting set operator and the set operator for the preceding proof step. This process is called *unification*. If the unification fails, the set operator is appended to the partial method which is called *division*. Then, the next proof step is processed. When all proof steps are processed, the partial method generates a set of proof steps containing the proof. The procedure contains time limits which are required because the partial methods are constructed on the basis of experiments which may fail.

The method acquisition procedure is an iteration procedure whose iteration variables are a *pointer* to the current proof step, i.e., the proof step that is presently being analyzed, a *partial method*, and a *partial proof*. A *partial method* is a list of set operators which have priority in reverse order, i.e., an operator is executed only if the following operators do not generate proof steps. The partial proof which is a list of proof steps generated by the partial method contains the proof steps up to the current proof step. At the beginning, the pointer is 1 and the partial method and the partial proof are the empty list. The iteration step, i.e., the analysis of a single proof step, has six stages:

Stage 1 (Application of the Partial Method). This stage applies the partial method within a time interval such as ten minutes. If it produces a set of proof steps containing the proof steps up to the current proof step, the sixth stage of the iteration step is performed. Otherwise, the second stage is performed.

Example (fourth proof step). The analysis of the first, second, and third proof step of the proof in Section 2 which is described in the examples of Stages 2 to 5 yields the method in Table 2. This method generates all proof steps, in particular the fourth proof step.

Stage 2 (Elementary Analysis). This stage produces a set operator

First operator	Second operator		
$g(a,a)b = g(g(a,a)a,a)b$			
...			
$g(a,a)b = g(a,a)(g(a,b)a)$	$g(a,a)(g(a,b)a) = (g(a,a)g(a,b))a$		
$g(a,a)b = g(a,a)(g(b,b)b)$	$g(a,a)(g(b,b)b) = (g(a,a)g(b,b))b$		
$g(a,a)b = g(ah(a,a),a)b$			
...			
$\underline{g(a,a)b = g(a,a)(ah(a,b))}$	$\underline{g(a,a)(ah(a,b)) = (g(a,a)a)h(a,b)}$	$\underline{(g(a,a)a)h(a,b) = ah(a,b)}$	$\underline{ah(a,b) = b}$
$\underline{g(a,a)b = g(a,a)(bh(b,b))}$	$g(a,a)(bh(b,b)) = (g(a,a)b)h(b,b)$		

Table 3: The equations generated by the proof method

$\{(e,t,p,a,s) |$ *is-a-proof-step* $(e,t,p,a,s) \wedge ...\}$

whose constraints are constructed as follows: The application of axioms in the knowledge base of SHUNYATA to initial metatheorems yields generations of new metatheorems. The initial metatheorems state that the theorem and the equation in a proof step are equations, that the term in a proof step is a term, that the axiom in a proof step is an equation, and that the substitution in a proof step is a substitution. The axioms contain elementary knowledge about the elementary functions that can occur in set operators such as descriptions of the domains and ranges and the fact that the application of connectives and predicates to valid arguments yields formulas. Thus, some metatheorems contain formulas. The formulas that are evaluable and whose value is true for the current proof step are used as additional constraints on the set operator. If the evaluation of the resulting set operator within a given time interval such as ten minutes yields a set of proof steps, the third stage of the iteration step is performed. Otherwise, the axioms in the knowledge base are again applied to the metatheorems which yields a new generation of metatheorems. This generation is processed as described.

Example (first proof step). The elementary analysis of the first proof step of the proof in Table 1 produces a set operator containing the constraints in the first operator in Table 2. As an example, the construction of the constraint

left-side (e) = *left-side* (*THEOREM*)

in this set operator is described. Initial metatheorems are

is-an-equation (*EQUATION*) and
is-an-equation (*THEOREM*),

which means that the equation in the current proof step and the theorem are equations. The application of the axiom

$\forall x$: *is-an-equation* $(x) \Rightarrow$ *is-a-term* (*left-side* (x)),

which describes the domain and the range of the function *left-side*, to the two initial metatheorems yields the metatheorems

is-a-term (*left-side* (*EQUATION*)) and
is-a-term (*left-side* (*THEOREM*)),

which state that the left side of the equation in a proof step and the left side of the theorem are terms. The application of the axiom

$\forall x, y$: *is-a-term* $(x) \wedge$ *is-a-term* $(y) \Rightarrow$ *is-a-formula* $(x = y)$,

which states the application of the equality predicate to two terms forms a formula, to the metatheorems yields the metatheorem

is-a-formula (*left-side* (*EQUATION*) =
left-side (*THEOREM*)).

Because the formula

left-side (*EQUATION*) = *left-side* (*THEOREM*)

in this metatheorem is evaluable and its value is true, the constant *EQUATION* is replaced by the variable *e* and the resulting formula is used as an additional constraint of the set operator.

Stage 3 (Simplification). The set operator generated by the preceding stage contains unessential constraints. Therefore, this stage multiplies the evaluation time of the operator by a number greater than one such as two, four, or eight which produces a time interval. Then, it temporarily removes each constraint from the set operator. If the evaluation of a resulting set operator produces the current proof step within the time interval, the constraint is deleted definitively and the next constraint is processed. Thus, a set operator is generated that contains only essential constraints. Then, the fourth stage of the iteration step is performed.

Example (first proof step). The set operator generated by the elementary analysis of the first proof step contains many unessential constraints such as

left-side (*THEOREM*) = *left-side* (*THEOREM*)

which means that the left side of the theorem is equal to itself. The deletion of unessential constraints yields the first operator in Table 2. It forms the partial method after the analysis of the first proof step.

Stage 4 (Unification). If the partial method is the empty list, the fifth stage of the iteration step is performed. Otherwise, this stage attempts to construct a set operator

$\{(e,t,p,a,s) |$ *is-a-proof-step* $(e,t,p,a,s) \wedge ...\}$

by unifying the set operator generated in the simplification stage and the last set operator in the partial method. The constraints of the set operator are constructed in two substages:

(a) Analogous to the elementary analysis in the second stage, generations of metatheorems are produced on the basis of initial metatheorems and axioms in the knowledge base of SHUNYATA. Metatheorems stating that the constraints in the two set operators to be unified are formulas are additionally used as initial metatheorems. The formulas in metatheorems that are evaluable and whose value is true for the proof step preceding the current proof step and the current proof step are used as additional constraints on the set operator. If the repeated evaluation of the resulting set operator within a time interval such as ten minutes yields the proof step preceding the current proof step and the current proof step itself, Substage (b) is performed. Otherwise, the next generation of metatheorems is processed as described.

(b) Analogous to the simplification stage, unessential constraints in the set operator produced in Substage (a) are deleted. Then, the last set operator in the partial method is replaced by the resulting set operator and the sixth stage of the iteration step is performed.

If the execution time of this stage is greater than the time required for the construction of the set operators to be unified, the fifth stage of the iteration step is performed.

Example (third proof step). The elementary analysis and the simplification stage for the third proof step yield the second operator in Table 2. The unification of the set operator generated by the analysis of the second proof step (see the example

in Stage 5) and the second operator yields the second operator because the value of the constraints

variables (*left-side* (*a*)) $\supseteq$ *variables* (*right-side* (*a*)) and

right-side (*e*) $\notin$ *left-sides* (*equations* (*PARTIAL-PROOF*))

in the second operator is true for the second proof step. The second operator in the partial method after the analysis of the second proof step is replaced by the second operator which yields the proof method in Section 3.

Stage 5 (Division). The set operator generated by the simplification stage is appended to the partial method. Then, the sixth stage of the iteration step is perfomed.

Example (second proof step). The elementary analysis and the simplification stage for the second proof step produce a set operator whose first constraints are the first three constraints in the first operator in Table 2 and whose last constraint is

substituents (*s*) = *variables* (*THEOREM*)

Because the unification fails, this operator is appended to the partial method generated by the analysis of the first proof step.

Stage 6 (Update of the Iteration Variables). If the pointer points to the last proof step, the partial method generates a set of proof steps containing a proof for the theorem. Otherwise, the remaining iteration variables are updated. The pointer to the current proof step is increased by one and the new partial proof is obtained by appending the proof steps generated by the new set operator to the old partial proof.

5 Application of the Proof Method

The application of simple variations of the proof method in Table 2 to other axiomatizations of group theory and certain theorems in equality yields proofs for these theorems. Examples are the theorems that there is only one identity element in a group, that the inverse element of an element is unique, and that the inverse element of the inverse element is equal to the original element (see [MacLane & Birkhoff 67, pp. 75-77]). An example of a variation of the proof method is the application of subterms of the theorem as substituents in its first operator. Variations of the proof method are also capable of generating proofs for sophisticated theorems such as SAM's Lemma whose complexity represents the state of the art in automated theorem proving (see [Antoniou & Ohlbach 83, p. 919]). Proofs of SAM's Lemma generated by traditional theorem provers are unreadable and manual translations of these proofs into readable form can be over a page long (see [Ohlbach 82, pp. 60-61]). SAM's Lemma is a theorem in equality which was an open problem in modular lattice theory until 1969. This section describes a variation of the method in Table 2 which generates a proof for SAM's Lemma. The proof is fairly readable and simpler than any other proof of SAM's Lemma known to the author.

The following axiomatization describes a modular lattice with a zero and a one element for which four additional axioms about constants a, b, c, and d hold. This axiomatization requires two binary functions f and g. We write $x+y$ for $f(x,y)$ and xy for $g(x,y)$, where x and y are terms. The axioms are:

1. $(x+y)+z = x+(y+z)$
2. $(xy)z = x(yz)$
3. $x+y = y+x$
4. $xy = yx$
5. $x+x = x$
6. $xx = x$
7. $x+xy = x$
8. $x(x+y) = x$
9. $x+z = z \Rightarrow (x+y)z = x+yz$
10. $0+x = x$
11. $0x = 0$
12. $1+x = 1$
13. $1x = x$
14. $(a+b)+c = 1$
15. $(a+b)c = 0$
16. $ab+d = 1$
17. $(ab)d = 0$

An example of such a lattice is a finite set with the union and the intersection of subsets as binary operators, the empty set and this finite set as a zero and a one element, c as the complement of the union of the sets a and b, and d as the complement of the intersection of the sets a and b. SAM's Lemma is

$$(c+da)(c+db) = c.$$

A proof method for SAM's Lemma is obtained by modifying the method in Table 2 as follows:

1. The last constraint in the first operator is replaced by
   substituents (*s*) $\subseteq$ *constants* (*THEOREM*)
   which means that the substituents in the substitution are constants in the theorem.
2. The constraint
   constants (*left-side* (*a*)) $\supseteq$ *constants* (*right-side* (*a*)),
   which means that the constants in the right side of an axiom in a proof step are contained in its left side, is added to the constraints of the second operator.
3. The consequent $(x+y)z = x+yz$ in the ninth axiom is applied only if the antecedent $x+z = z$ can be proved by the second operator.

The application of this variation of the proof method in Table 2 to SAM's Lemma yields some 86,000 proof steps nineteen of which form a proof for SAM's Lemma [Ammon 87]. By omitting parentheses, the number of proof steps can be drastically reduced: from 86,000 to just 111. (Humans also omit the parentheses for nested associative-commutative functions such as addition.) Eight of the 111 proof steps form the proof in Table 4. In each case, the dots represent the right side of the preceding equation. Variations of a given method can be generated by a simple *trial-and-error* procedure: The method acquisition procedure is applied to some simple proofs and constraints in the set operators of the resulting methods are tentatively inserted into the set operators of the given method until a successful variation is generated.

6 Discussion

If the order of the axioms in the knowledge base of SHUNYATA is reversed, the method acquisition procedure constructs the proof method in Table 2 after the analysis of the second proof step so that no unification is performed. This run of SHUNYATA came as a surprise to the author, who thought a unification was required. The application of the method acquisition procedure to the proof obtained from the proof in Section 2 by exchanging the left and right sides of the equations and reversing the order of the proof steps yields a method which is similar to the method in Table 2. If in the simplification stage for the third proof step the evaluation time of the set operator is multiplied by eight, the constraint

right-side (*e*) $\notin$ *left-sides* (*equations* (*PARTIAL-PROOF*))

is deleted, which means that the system regards the constraint as unessential. This was another surprise to the author, who thought that the constraint could not be omitted. In this case, the author learned an interesting fact from the SHUNYATA program. The application of the method acquisition procedure to proofs for other theorems in group theory also yields useful proof methods [Ammon 87]. SHUNYATA discovered proofs for significant theorems from higher mathematics such as the fixed point theorem on the basis of methods which have so far been developed manually. The methods consist of simple operators generating finite sets of proof steps and simple heuristic rules controlling their application. They could also be constructed by applying the method acquisition procedure to simple proofs [Ammon 87, 88]. The discovery of a proof for the fixed point theorem is the first discovery of a proof for a significant theorem from higher mathematics by a machine (see [Wos 86]). The method acquisition procedure is also capable of constructing visual concepts from preprocessed images

Equation		Axiom	Substitution
$(c+da)(c+db)$	$=(c+da(a+b))(c+db)$	8	$\{a/x,\ b/y\}$
...	$=c+da(a+b)(c+db)$	9	$\{c/x,\ da(a+b)/y,\ c+db/z\}$
...	$=c+da(db+c(a+b))$	9	$\{db/x,\ c/y,\ a+b/z\}$
...	$=c+da(db+0)$	15	$\emptyset$
...	$=c+dadb$	10	$\{db/x\}$
...	$=c+d0$	17	$\emptyset$
...	$=c+0$	11	$\{d/x\}$
...	$=c$	10	$\{c/x\}$

Table 4: A proof of SAM's Lemma generated by the SHUNYATA program

[Ammon & Stier 88]. These examples show the generality and power of the method acquisition procedure.

Why does the method acquisition procedure work? The experiments with SHUNYAYA suggest that mathematical cognition is surprisingly shallow. For example, using simple variations of the proof method in Table 2, one can generate proofs that cover more than a page in standard algebra textbooks (see Section 5). The variation described in Section 5 even generates a proof for SAM's Lemma, whose complexity represents the state of the art in automatic theorem proving. The proof method itself is simple and can automatically be constructed by an analysis of a simple proof or even two proof steps (see preceding paragraph). The method acquisition procedure is a simple embodiment of a general knowledge acquisition procedure described in [Ammon 87]. It can be outlined as follows: Elementary knowledge about elementary functions forms the basis of knowledge acquisition processes. This knowledge includes axioms describing the domains and ranges of the elementary functions and axioms stating that the application of connectives or predicates to valid arguments yields formulas such as the axiom that the application of the equality relation to two terms yields a formula. This knowledge is represented in the functional knowledge representation language CL [Ammon 87]. Thus, CL provides elementary building blocks for the automatic construction of new concepts and methods such as visual concepts and proof methods. The acquisition of knowledge is regarded as an inference process which applies the elementary knowledge to new information such as an image or a proof and thus generates a sequence of more and more powerful propositions which, for example, contain partial concepts or partial methods. The propositions are evaluated and the true propositions are *fed back* into the knowledge acquisition process and used for further inferences. This means that the development of new knowledge is genetically controlled, i.e., new knowledge arises from elementary analyses of new information on the basis of more elementary knowledge and is repeatedly unified until sophisticated knowledge is generated. The elementary analyses can be regarded as division processes. Thus, new knowledge arises from more elementary knowledge by divisions and unifications. Because new knowledge is not completely determined by more elementary knowledge, variations of new knowledge are rather randomly generated, tested, and the variations that prove to be successful are fed back into the knowledge acquisition process and the other ones are abandoned. The generality of the knowledge acquisition procedure is due to the universality of its basis because there are limited sets of elementary knowledge about elementary functions from which extensive sets of sophisticated knowledge can be composed. For example, there are universal sets of elementary computable functions from which all computable functions can be composed.

SHUNYATA is written in LISP and is presently running on a Symbolics 3600 and an IBM PC AT. It consists of some 5,000 lines of source code. The analyses of proofs take from several hours up to several days. The 111-step generation of the proof of SAM's Lemma discussed in Section 5 took approximately seventy-five minutes. Because SHUNYATA is an experimental system, its code is not optimized. Recent tests indicate that an improvement of its time efficiency by a factor of ten is feasible.

7 Related Work

Lenat's AM program discovered mathematical definitions and conjectures on the basis of a set of initial concepts and a large number of sophisticated heuristics [Lenat 82, pp. 35-101 and pp. 151-204]. In contrast, SHUNYATA constructs proof methods on the basis of elementary knowledge about elementary functions such as descriptions of their domains and ranges. AM was the first project concerned with automated mathematics research [Lenat 82, p. 137]. AM is concerned with elementary mathematics [Lenat 82, p. 7] whereas SHUNYATA focuses on higher mathematics. This comparison also applies to Lenat's EURISKO program [Lenat 83].

Silver's LP program learns new equation-solving methods by a sophisticated technique which is called precondition analysis [Silver 86]. In contrast, SHUNYATA constructs proof methods on the basis of elementary functions.

Mitchell *et al.* [86] describe a learning mechanism called explanation-based generalization. This mechanism transforms an inefficient definition of a goal concept into an efficient definition on the basis of a description of a training example, a domain theory, and an operationality criterion [Mitchell *et al.* 86, pp. 50-52]. The goal concept defines the concept to be acquired, the training example is a description of an example, and the operationality criterion defines efficient terms in which the goal concept must be expressed. In contrast, SHUNYATA does not contain the goal concept at the beginning, but only elementary functions that form the building blocks of methods. Furthermore, it contains the proof, i.e., the training example, explicitly. The axioms containing elementary knowledge about the elementary functions could be regarded as an elementary domain theory. Finally, SHUNYATA does not contain an operationality criterion but it selects efficient constraints in the simplification stage of the method acquisition procedure on the basis of experiments.

8 Conclusion

This paper described the method acquisition processes in the SHUNYATA program which analyzes mathematical proofs and constructs methods capable of generating proofs of new theorems. The methods are lists of set operators which are composed of elementary functions. Elementary knowledge about these functions such as descriptions of their domains and ranges forms the basis of the method acquisition processes. In the analysis of proofs, SHUNYATA processes the proof steps one by one and, starting from scratch, constructs a sequence of more and more powerful partial methods until a complete proof method is generated. In the analysis of a single proof step, it

$\forall e, t, p, a, s:$	$is\text{-}a\text{-}proof\text{-}step\ (e,t,p,a,s) \Longleftrightarrow$ $is\text{-}an\text{-}equation\ (e) \wedge is\text{-}a\text{-}term\ (t) \wedge is\text{-}a\text{-}pointer\ (p) \wedge is\text{-}an\text{-}axiom\ (a) \wedge is\text{-}a\text{-}substitution\ (s) \wedge$ $left\text{-}side\ (e) = t \wedge right\text{-}side\ (e) = replace\ (t, p, right\text{-}side\ (substitute\ (a,s))) \wedge p \in pointers\ (t) \wedge$ $arg\ (p,t) = left\text{-}side\ (substitute\ (a,s))$
$\forall e, t, p, a, s:$	$is\text{-}a\text{-}proof\text{-}step\ (e,t,p,a,s) \wedge variables\ (left\text{-}side\ (a)) \supseteq variables\ (right\text{-}side\ (a)) \Longrightarrow$ $s = match\ (arg\ (p,t),\ left\text{-}side\ (a))$

Table 5: Two axioms required for the evaluation of the second operator of the proof method

$\{(e,t,p,a,s) \mid$	$is\text{-}a\text{-}proof\text{-}step\ (e,t,p,a,s) \wedge \ldots \wedge right\text{-}side\ (e) \notin left\text{-}sides\ (equations\ (PARTIAL\text{-}PROOF))$ $is\text{-}an\text{-}equation\ (e) \wedge is\text{-}a\text{-}term\ (t) \wedge \ldots \wedge arg\ (p,t) = left\text{-}side\ (substitute\ (a,s)) \wedge$ $s = match\ (arg\ (p,t),\ left\text{-}side\ (a))$

Table 6: The extended second operator of the proof method

first performs an elementary analysis which yields a set operator, then simplifies this operator, finally attempts to unify the resulting operator and the partial method, or inserts the resulting operator into the partial method if this attempt fails. A final objective of my work is the development of a program that automatically analyzes mathematics textbooks and automatically develops new mathematical theories in ordinary representation. Future experiments will for example deal with the automatic acquisition of powerful proof methods generating proofs for significant theorems from higher mathematics.

Acknowledgements

I would like to thank David Fleet and the referees for providing valuable comments on an earlier draft of this paper. Special thanks to Russell Block for improving my English.

Appendix

The operators in the proof method in Table 2 generate the set of all proof steps that satisfy certain constraints. Their evaluation is performed in two steps. The first step applies axioms about proof steps in the knowledge base to the constraints on a set operator which yields additional constraints. The second step replaces element relations between parts of proof steps and finite sets by equality relations between these parts and the elements of these sets. If the constraints of a resulting set operator are inconsistent, this operator is abandoned. Otherwise, it is used for generating a proof step. For example, the evaluation of the second operator of the proof method is performed as follows: The knowledge base of SHUNYATA contains the two axioms in Table 5. The first axiom is a definition of the concept of proof steps. It states that the first member e of a proof step is an equation, its second member t is a term, its third member p is a pointer to a subterm of t, its fourth member a is an axiom, its fifth member s is a substitution, the the left side of the equation e is equal to the term t, and so forth. The *replace* function replaces the subterm of a term a pointer points to by another term. The *pointers* function generates the set of all pointers that point to the subterms of a term. The *arg* function selects the subterm of a term a pointer points to. The second axiom states that, if the variables in the right side of an axiom a in a proof step (e,t,p,a,s) are contained in its left side, then the substitution s is obtained by matching the left side of the axiom a against the subterm of the term t the pointer p points to. Thus, this axiom describes a simple property of special proof steps. The first step of the evaluation applies the two axioms to the second operator which yields the operator in Table 6. The second step of the evaluation successively replaces the three element relations by equality relations between p, *left-side* (e), and a and the corresponding finite sets. If the constraints in a resulting set operator are inconsistent, this operator is abandoned. Otherwise, the left side and the right side of the equation e, the term t, the pointer p, the axiom a, and the substitution s are determined by equality relations contained in this operator. Therefore, a proof step can be constructed from these parts directly. Thus, the set of all proof steps that satisfy the constraints in the original set operator is generated.

References

[Ammon 87] K. Ammon. *The Automatic Development of Concepts and Methods.* Doctoral dissertation, Department of Computer Science, University of Hamburg, 1987.

[Ammon 88] K. Ammon. Discovering a proof for the fixed point theorem: A case study. In Y. Kodratoff (Ed.) *Proceedings of the Eighth European Conference on Artificial Intelligence*, Munich, August 1988. Pitman, London, 1988.

[Ammon & Stier 88] K. Ammon and S. Stier. Constructing polygon concepts from line drawings. In Y. Kodratoff (Ed.) *Proceedings of the Eighth European Conference on Artificial Intelligence*, Munich, August 1988. Pitman, London, 1988.

[Antoniou & Ohlbach 83] G. Antoniou and H. J. Ohlbach. Terminator. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, August 1983. Kaufmann, Los Altos, 1983.

[Lenat 82] D. B. Lenat. AM: Discovery in mathematics as heuristic search. In R. Davis and D. B. Lenat, *Kowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, New York, 1982.

[Lenat 83] D. B. Lenat. EURISKO: A program that learns new heuristics and domain concepts. *Artificial Intelligence*, Vol. 21, 1983, pp. 61-98.

[MacLane & Birkhoff 67] S. MacLane and G. Birkhoff. *Algebra.* Macmillan, New York, 1967.

[Mitchell *et al.* 86] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, Vol. 1, 1986, pp. 47-80.

[Ohlbach 82] H. J. Ohlbach. The Markgraf Karl refutation procedure: The logic engine. Interner Bericht 24/82, University of Karlsruhe, Karlsruhe, West Germany, 1982.

[Silver 86] B. Silver. Precondition analysis: Learning control information. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.) *Machine Learning: An Artificial Intelligence Approach*, Vol. II, Morgan Kaufmann, Los Altos, 1986.

[Wos 86] L. Wos. From the president. *Newsletter of the Association for Automated Reasoning*, No. 7, 1986, p. 1.