TR-140

An Algebraic Manipulation System Using Meta-level Inference Based on Human Heuristics

by
Toshiaki Takewaki, Taizo Miyachi
Susumu kunifuji and Koichi Furukawa

October, 1985

An Algebraic Manipulation System Using Meta-level Inference Based on Human Heuristics

by
Toshiaki Takewaki , Taizo Miyachi
Susumu Kunifuji , Koichi Furukawa

ICOT Research Center
Institute for New Generation Computer Technology

ABSTRACT

This paper describes an algebraic manipulation system, AMIE*, in which meta-level inference is used to implement human heuristics. Algebraic manipulation systems are expert systems to solve algebraic problems. Not only AMIE solves equations of elementary functions in one variable, it also differentiates and integrates one-variable elementary functions. The characteristic of AMIE is solving algebraic problems in the same way as human do. Human problem-solving techniques are represented in meta-level knowledge and meta-level inference that simulates human problem solving.

In general, user interface is very important in an expert system. AMIE has two facilities for user interface: the first, it explains the process of problem solving on displays, the second, it can evaluate the relative degree of problem-solving-difficulty for educational use. AMIE is written in Prolog, therefore, meta-level inference and knowledge can be implemented straightforwardly. We have also noticed that the meta-level inference enables the system to be modified with ease.

1. Introduction

Algebraic manipulation systems are problem-solving system which uses algebraic knowledge. In general, several stages should be explored for solving algebraic problems: recognition of problems, analysis for problem solving, selection of strategies, break-down into applicable methods, and an application of methods. Each stage needs human heuristics accumulated for long years. In algebraic manipulation systems, the heuristics are described in rewriting rules or control-knowledge for applying the rules. Therefore, the system should prepare system-functions for simple expression and modification of knowledge for controls as well as rewriting rules.

The following research has been done: SAINT [Slagle 61] is the first algebraic manipulation system that selects the rule based on many heuristics as A.I. approach. SCRATCHPAD [Jenks 84] has been built using simple rewriting rules. However, it lacks concepts of a substitution and a procedure. As practical more systems, REDUCE [Hearn 84], and MACSYMA [Mathlab 77] are well known. PRESS [Bundy 81, Sterling 82] introduces object inference and meta inference, but, distinctions between object level knowledge and meta level knowledge could not be expressed clearly in programming methods.

* an Algebraic manipulation system using Meta InferencE

AMIE incorporates several functions useful for each problem solving process. One is a substitution function to simplify a formula substituting a variable for sub-expression including variables. This function can be easily implemented using the unification function in Prolog. If system designers only must give proper structures into object data of unification corresponding to given application, the reasoning functions of Prolog with unification function supports a substitution function. AMIE makes it easy for users to express distinction between the object-level worlds and the meta-level worlds, due to easiness in programming of in prolog [Bowen 81, Furukawa 84, Miyachi 84]. This representation enables users easily understand and express of object-knowledge and meta knowledge, the inside of system structure. Adding and changing of the system functions are very easy using a meta programming method. Controls of strategies for problem solving are also easily described using the meta-programming method.

Control functions for meta inference in AMIE is based on human way in problem solving. That it, the system can take the same approach as human does in solving algebraic expression and new heuristics can be easily added into the system. The system has two functions for users: one function is to help human understanding of the process in problem solving and the other is to define and express the relative degrees of difficulty involved in the problem solving. Users can know rough degrees of effectiveness for given problems, looking at the degrees given by the latter function.

AMIE was developed to solve elementary functional equation, such as that in differential and integral calculus with a single occurrence of a variable of the kind introduced in high school and university text books. Some typical problems are shown in Figure 1-1.

In this paper, Chapter 2 describes the main objective of AMIE and chapter 3 a canonical form of an algebraic expression and a simplification function. Chapter 4is a control methods of algebraic manipulation such as a solving equation and integral, and Chapter 5 an inference method based on human heuristics. Chapter 6 gives definition and expression of relative degrees of difficulty in problem solving with an example of equation solving in chapter 7.

$$4X^4 - 17X^2 + 4 = 0 \qquad \text{(Expression 1)}$$

$$2\log_2 X - 3\log_X 2 + 5 = 0 \qquad \text{(Expression 2)}$$

$$3\cos^2 X + 5\sin X - 1 = 0 \qquad \text{(Expression 3)}$$

$$\left(\log|X + \sqrt{X^2 + A}|\right)' \qquad \text{(Expression 4)}$$

$$\int X * e^{-X^2} dX \qquad \text{(Expression 5)}$$

$$\int X^2 * arcsin\, X dX \qquad \text{(Expression 6)}$$

Figure 1-1 Some typical problems handled by the program

2. System Characteristics

The main objective of AMIE is to solve algebraic in the same way as a human solves them and to construct a system that helps humans understand the process of problem solving. The system is intended, to clarify the inference process rather than to provide a capability for high-speed processing. The method used in building up the stages of inference is exactly the same as the reasoning processes and methods human beings normally uses in solving algebraic expressions.

The characteristics of AMIE are:
(1) Introduction of meta-level inference
(2) Introduction of degrees of difficulty
(3) Introduction of explanation function
(4) Use of Prolog.

Each of these characteristics are described below in detail.

2.1 Meta-level Inference

The unlimited use of various rules for mathematical problem solving will result in an explosion of computation volume because of the over extended use of search space. Consequently it becomes extremely difficult to solve the problem in real time. Furthermore, it cannot be generally said that the search is oriented always in the optimum direction. There is, therefore, a need to minimize wasteful search by a selective use of search space and directing search in what is believed to be the best direction. Hence the *demo* predicate, which is in fact a meta-predicate, and the introduction of meta-level inference for search control. These two features made possible these describable characteristics.

(1) Easy determination of complex processing control.

(2) Modular approach to each process rule making additions and changes easy.

(3) Reduction of wasteful search space by effective selection of search space.

(4) Clear separation of meta- and object-level inference.

(5) Clear distinction of the categories of control and structure for each problem type.

2.2 Inference Process and Degrees of Difficulty

An attempt was made to express degrees of difficulty to the inference process. This was achieved by adding degrees of difficulty based on human empirical knowledge. Among those for which this program was developed, in some cases different kinds of mathematical problems can be solved by cases by more than one method. Different methods can be applied depending on the degree of difficulty of the problem. Relative degrees of difficulty were defined by giving a weighted cost to each rewriting rule. This enables the selection and use of the least costly rewriting rule and thereby contributes to the efficiency of the process. Furthermore, relative degrees of difficulty are defined for different types of equations.

2.3 Explanation Function

When a person is involved in solving algebraic equations, he normally jots down the process of computation for the more complex problems. By doing this he shows how the problem was solved. The same process is expressed in our program by the use of the explanation function. The program shows the user what solution method was applied and explains that method.

The program also traces the "notes" for each computation process just as a person can refer to his notes afterwards. This extra function helps the user to comprehend the reasons for the rules selected and applied and he will learn to manipulate expression transformation . Multiple windows are used for this function to help the user understand the process more readily.

2.4 Use of Prolog

Prolog was chosen because it is a logic language. The choice was made because we believed the basic function of Prolog is suited to algebraic manipulation. The following characteristics were obtained.

(1) Easier meta-level control.

(2) When multiple solution methods are available, the Prolog backtrack function provided automatic use of other solution methods.

(3) Prolog syntax made the description of rewriting rules for algebraic manipulation easy.

(4) Easy pattern matching for application of rewriting rules.

3. Algebraic Expression and Simplification Method

3.1 Canonical Form and Powerful Matching Function

An algebraic expression is composed of function(s), variable(s), constant(s) and operator(s). An attempt to solve the expression in its original form as input into the system will make the process complex and cumbersome due to the large number of rewriting rules. The number of rewriting rules, storage capacity, and processing speed are all affected by the way the expression is written for the system and by the way the patterns are matched. The system addresses this problem by employing two standardized types for algebraic expression and a powerful matching function which considers commutation This process greatly reduces the number of rewriting rules needed and substantially simplifies the algebraic expression resulting in efficient processing.

(1) Canonical Form to Reduce Number of Rewriting Rules

Generally speaking, the more operators, the more rewriting rules are needed. To reduce the number of rewrites required the following transformations were applied to deduction (Rule 1) and division (Rule 2) respectively. So the four computation rules are then expressed by two rules for addition and multiplication.

$$X - Y \quad \Rightarrow \quad X + (-1) * Y \qquad \text{(Rule 1)}$$

$$X / Y \quad \Rightarrow \quad X * Y^{-1} \qquad \text{(Rule 2)}$$

Without changing the form of the expressions, expressions 6 and 7 can be rewritten by the use of rules 3 and 4 respectively. When the expression is transformed using Rule 1 and 2, then expressions 6 and 7 can be processed using the same rewriting rule 3.

$3*X + 4*X$			(Expression 7)
$3*X - 4*X$			(Expression 8)
$A*X + B*X$	$\Rightarrow$	$(A + B)*X$	(Rule 3)
$A*X - B*X$	$\Rightarrow$	$(A - B)*X$	(Rule 4)

Expression 8 is rewritten using Rule 1 to $3*X + (-4)*X$. Now applying Rule 3, we get $(3 + (-4))*X$. The process effectively eliminates using Rule 4. What we have done is greatly reduce the number of rewriting rules by the use of Rule 1 and 2.

(2) Canonical Form of Polynomials

Algebraic expressions solved by AMIE are mainly those composed of the four arithmetic operations (exponentiation is expressed by multiplication). These can be expressed as polynomial or rational expressions. In turn the polynomials and rational expressions may be considered as follows:

(a) A rational expression may be expressed as a ratio of polynomial.

(b) A polynomial may be expressed as the sum of terms.

(c) A term may be expressed as coefficient and degree of the main variable.

From the canonical form of the polynomial:

Polynomial

$$a_0X^n + a_1X^{n-1} + \dots + a_n \qquad (a_0 \neq 0)$$

we can write

$$[(a_0, n), (a_1, n-1), \dots, (a_n, 0)].$$

Similarly for rational expressions:

$$\frac{a_0X^n + a_1X^{n-1} + \dots + a_n}{b_0X^m + b_1X^{m-1} + \dots + b_m} \qquad (a_0 \neq 0, b_0 \neq 0)$$

can be written as

$$[([(a_0, n), (a_1, n-1), \dots, (a_n, 0)], [(b_0, m), (b_1, m-1), \dots, (b_m, 0)])].$$

The use of such internal expression enables the system to understand that $(X^2+1)(X+1)$ and $X^3 + X^2 + X + 1$ are identical expressions. The process eliminates complex pattern matching required for applying rules and contributes to efficient processing.

(3) Powerful Matching Function

The expressions $\sin X * \cos X$ and $\cos X * \sin X$ cannot be matched using the matching function inherent in Prolog. We developed a powerful matching function has been introduced by considering neutral elements and rules of exchange for multiplication and addition. This

Rules			Prolog syntax
X^1	$\Rightarrow$	X	$simplify(X^1, X).$
$1{*}X$	$\Rightarrow$	X	$simplify(1{*}X, X).$
$X+0$	$\Rightarrow$	X	$simplify(X+0, X).$

Figure 3-1 Rules for Simplification

```
simplify(X+Y+Z,S+Z):-rational(X),
                     rational(Y),
                     S := X+Y.
```

Note: rational is a predicate that judges rational numbers and ':=' performs the four arithmetic operations for rational numbers.

Figure 3-2 Examples of Simplification

matching function uses Rule 5 through 8 and eliminates application of similar rewriting rules reducing them to the most general and the fewest.

$A + B = B + A$ (Rule 5)

$A{*}B = B{*}A$ (Rule 6)

$1{*}A = A$ (Rule 7)

$A + 0 = A$ (Rule 8)

3.2 Simplification of Algebraic Expressions

In the process of solving algebraic expressions, the repeated use of rules without a reduction process will result in a long expression because of the explosive number of intermediate solutions. Such an overloaded expression will invite confusion and consume large amounts of memory and CPU capacity. AMIE reduces the expression every time a rule is applied to check the expression from becoming too long. Broadly speaking there are two types of reduction processes used: (1) exclusion of unnecessary coefficients and degrees, and (2) compound.

(1) Figure 3-1 shows two rules for eliminating unnecessary coefficients and degrees.

(2) The other rule is for compounding constants using the four arithmetic operations for rational numbers. For example, $\frac{1}{2} + \frac{1}{3} + X$ may be written as $\frac{5}{6} + X$ by compounding $\frac{1}{2}$ and $\frac{1}{3}$. This can be written using Prolog as Figure 3-2 shows.

4. Methods of Control

In attempting to solve algebraic expressions humans find certain characteristics of the expression. AMIE follows the same pattern by abstracting various characteristics found in the given expression. Using the abstracted characteristics the program uses inference to find applicable rules. This inference process is conducted by meta-level inference which can be broadly divided into form control and rule control. The form control function classifies the expression according to its form. The rule control function selects applicable rules based on characteristics other than the form. The control program is shown in Figure 4-1.

The predicate *'manipulation'* solves expression *'Exp'* for variable *'Var'* and returns the answer *'Ans'*. When control information is described in *'Ctrl'*, it uses the information as a priority to solve the expression. The program also records in *'History'* all the rewriting rules applied before the solution is found and returns to *'Cost'* the actual degree of difficulty experienced.

The predicates *'expression _ type'* and *'exp _ type _ manipulation'* conduct form controls. Whereas the predicate *'expression _ type'* judges the form of the expression, the predicate *'exp _ type _ manipulation'* conducts integral and transform processing according to the given form.

The predicate *'method _ demo'* directs rule control to solve the expression *'NewExp'* with variable *'Var'*. The details of this process are described later.

The predicate *'difficulty'* computes the degree of difficulty of the process involved from the history of rewriting rules.

Meta-level inference using a dictionary of methods is in a effect control method making additions and changes easy.

4.1 Form Control of Equations

Form control classifies the equation by the location of the variables it contains into six form: polynomial, irrational, fractional, exponential, logarithmic, and trigonometric. A given equation may have several forms with various solution methods available. The structure of the method dictionary possibilities for application to learning systems.

(1) Polynomial Equation

A polynomial equation is expressed with polynomials on both sides of the equal sign:

$$a_0X^n + a_1X^{n-1} + ... + a_n = b_0X^m + b_1X^{m-1} + ... + b^m$$

The following is an example of a polynomial equation:

$$3*X^3 + \frac{1}{4}*X^2 + 5*X + 8 = 0$$

(2) Irrational Equation

Irrational equations are those containing variables within roots.

$$\sqrt{x-20} - \sqrt{x+1} = 0$$

```
manipulation(Exp,Var,Ctrl,Ans,Cost,History) :-
            expression_type(Exp,Var,ExpType),
            exp_type_manipulation(ExpType,Exp,Ctrl,NewExp,NewCtrl), !
            method_demo(NewExp,[Var,[],NewCtrl],Ans,[],History),
            difficulty(History,Cost).
```

Figure 4-1 Predicate 'manipulation'

(3) Fractional Equation

In fractional equations both sides are rational expressions and by rewriting the terms in standard rational expressions, the equation can be expressed as a polynomial equation with certain conditions attached. The following in an example of this type.

$$\frac{X+4}{X^2+X-2} = \frac{1}{X-1} + \frac{X+6}{X^2-4}$$

(4) Exponential Equation

This type of equation contains variables within the exponents. By substituting Y^n for a^{n*X} it can be transformed into a polynomial equation. The following is an example.

$$3*e^{2*X} + 4*e^X - 7 = 0$$

(5) Logarithmic Equation

Logarithmic equation contain variables within logarithms. By using basic transformations all bases are made equal before $\log X$ is substituted for Y to write a polynomial equation. The following in an example of this type of equation.

$$2*\log_2 X - 3*\log_X 2 + 5 = 0$$

(6) Trigonometric Equation

Variables in this type of equation are subject to trigonometric functions. By using the various trigonometric theorems, the equation may be transformed into one trigonometric function and rewritten as a polynomial equation. The following is an example.

$$3*\cos^2 X + \sin X - 1 = 0$$

4.2 Form Control of Integration

Form control of integration classifies the input expression by the type of the factors contains into one of a few types. The classification tree is shown in Figure 4.1. For example,in $\int 2*\sin X*X\ dX$,list least factors are [2,$\sin X$,X]. The input expression has three factors; 2 is constant type,$\sin X$ is sine type, and X is monomial type. The polynomial type contains the monomial type. The monomial type contains the constant type. The transcendental function type consists of exponential function type, logarithm function type , trigonometric function type, and so on.

4.3 Rules Control

This function is used to select applicable rules from the non-form characteristics of the given algebraic expression. There are various types of rules and when several rules are applicable the sequence described below is used. Figure 4-2 shows the predicate *'method _ demo'* used in this control. *'method _ demo'* indicates the expression input and various other control data (variable, information on applicable rule(s) given by user or available from past history (called next processing information) and other data). It also gives solutions to the expression, the history of rules used in the past, and updated history of rule usage after the current processing is completed.

MD1) checks for loop in the rewriting of the expression by checking and finding from history for identical problems. If the analysis is positive, a loop message is returned and operation is suspended.

MD2) breaks down the problem into its component expressions.

MD3) judges whether the expression was solved. The judgment is made by checking the restraints contained in the control. For example, give the request to solve equation $X^2 - 4 = 0$, the answers are $X = 2$ and $X = -2$. But if a condition $X > 0$ is stipulated, then $X = -2$ will not be a solution.

MD4) determines the applicable rule by the use of predicate *'strategy'* and applies the

```
Function
    |
    |--- Polynomial
    |          |
    |          |--- Monomial
    |                   |
    |                   |--- Constant
    |                            |
    |                            |--- Atom
    |                            |
    |                            |--- Number
    |                            |
    |
    |--- Transcendental function
    |          |
    |          |--- Exponential function
    |          |
    |          |--- Logarithmic function
    |          |
    |          |--- Trigonometric function
    |          |                  |
    |          |                  |--- Sine
    |          |                  |
    |          |                  |--- Cosine
    |          |                  |
    |          |                  |--- Tangent
    |          |                  |
    |          |
    |          |--- Inverse trigonometric function
    |          |                  |
    |          |                  |--- Arc sine
    |          |                  |
    |          |                  |--- Arc cosine
    |          |                  |
    |          |                  |--- Arc tangent
    |          |                  |
```

Figure 4-2 Classification Tree of Function Types

rule by the use predicate *'apply'*. Predicate *'explanation'* will have the system explain the reason for rewrite and predicate *'method _ demo'* processes the expression in regression.

4.4 Rewriting Rules for Equations

(1) Formula Rule

This rule is used when the equation can be solved with this rule without using other rules. This rule provides efficient processing by using basic formulas. An example is the formula for solving quadratic equations:

$$a*X^2 + b*X + C = 0 \qquad \Rightarrow \qquad X = \frac{-b \pm \sqrt{b^2 - 4*a*c}}{2*a}$$

(2) Isolation Rule

This rule is applied when there is only one variable in an equation. By applying inverse functions to both sides of the equation, functions surrounding a variable may be eliminated and isolated to one side of the equation. Some examples of this rule are given below:

$$\begin{array}{lcl} V + U = W & \Rightarrow & V = W - U \\ \log_U V = W & \Rightarrow & V = U^W \\ V^2 = W & \Rightarrow & [V = \sqrt{W}, V = -\sqrt{W}] \end{array}$$

(3) Collection Rule

```
MD1)  method_demo(Exp,[_,_,_],loop,History,loop) :-
              loop_check(History).

MD2)  method_demo(E1 && E2,[Var,Next,Ctrl],Ans1 && Ans2,History,Trace) :-
              method_demo(E1,[Var,Next,Ctrl],Ans1,History,Trace1),
              method_demo(E2,[Var,Next,Ctrl],Ans2,History,Trace2),
              append(Trace1,Trace2,Trace).

MD3)  method_demo(Exp,[_,end,Ctrl],Ans,_,[]) :-
              end_check(Ctrl,Exp,Ans), !.

MD4)  method_demo(Exp,[Var,Next,Ctrl],Ans,OldHis,Trace) :-
              strategy(Exp,World,[Next,Var,Ctrl],[I_Next,I_Var,I_Ctrl]),
              apply(World,solve(Exp,[I_Next,I_Var,I_Ctrl],I_Exp,
                              [N_Var,Explanation,H1,N_Next,N_Ctrl],_),
              explanation(H1,N_His,[N_Var,Explanation],Exp,I_Exp),
              method_demo(I_Exp,[N_Var,N_Next,N_Ctrl],Ans,
                              [N_His|OldHis],Trace1),
              append(Trace1,N_His,Trace).
```

Figure 4-3 Predicate 'method _ demo'

The purpose of collection is to reduce the number of variables to one in an equation or formula with two variables (not two types of variables). The expression to which this rule is applied does not have to be a complete equation. The collection rules are used to rewrite expressions in such a way that isolation can be applied. Some typical collection rules are

$$\begin{array}{lcl} \log_U V + \log_U W & \Rightarrow & \log_U V*W \\ \sin U*\cos U & \Rightarrow & \frac{1}{2}*\sin(2*U) \\ U*V + U*W & \Rightarrow & U*(V+W) \end{array}$$

all of which collect relative to U.

(4) Attraction Rule

The attraction rule will not reduce the number of variables as the collection rules do, but they can be applied to reduce the distance between two variables (not two types of variables). The reduction of distance between two variables here means that the unknowns come closer together so that they can be collected. The right hand side of the attraction rule has less distance between the variables compared to that on the left hand side. Some typical attraction rules are:

$$\begin{array}{lcl} \log_W U \log_W V & \Rightarrow & \log_W U + V \\ W*U + W*V & \Rightarrow & W*(U+V) \\ (W^U)^V & \Rightarrow & W^{U*V} \\ U^{V*W} & \Rightarrow & (U^V)^W \end{array}$$

which attract U and V.

(5) Substitution Rule

This rule is applied to equations of the form $h(f(X)) = g(f(X))$ or $h(f(X)) = 0$. Most of the trigonometric and exponential functions may be solved as general polynomial equation by substitution of unknowns. For instance, sub-expression $f(X)$ containing unknown X may be replaced by a new unknown variable. After solving the new equation the unknown will be substituted back in its original form and the original equation can then be solved. For example, in the equation $X^4 + 2*X^2 - 8 = 0$, $f(X) = X^2$ gives $(X^2)^2 + 2*(X^2) - 8 = 0$. Now substituting a new unknown Y for X^2 the equation can be rewritten as $Y^2 + 2*Y - 8 = 0$. Similarly $\sin^2 X + 2*\sin X - 8 = 0$ may be rewritten as $Y^2 + 2*Y - 8 = 0$ with $f(X) = \sin X$ and substituting Y for $\sin X$. Now the equation can be solved.

(6) Transformation Rule

This rule is applied in the following cases using the forms of the equations according to form control. After applying a transformation rule, however, the equation becomes conditional.

(a) Processing Logarithmic Equation

Transformation rules are applied when the equation contains logarithmic functions and when such functions do not have the same base. Only by the application of basic transformation rule can they share the same base.

For example, in $\log_2 X + 8*\log_X 2 - 6 = 0$, the logarithmic functions are $\log_2 X$ and $\log_X 2$. They do not have the same base, the latter base is an unknown. In order to eliminate the unknown contained in the base, rewriting rule $\log_X Y \Rightarrow \frac{1}{\log_Y X}$ is used for the transformation. The new equation may be written as $\log_2 X + 8/\log_2 X - 6 = 0$,which is soluble by the rule of substitution.

(b) Processing Fractional Equations

In this case the rule is applied by transforming the equation into a ratio of expressions. Then with the condition that the denominator is not zero, we solve for the nominator. Using standard form of the fractional equation $\frac{a_0 X^n + \ldots + a_n}{b_0 X^m + \ldots + b_m} = 0$, solve $a_0 X^n + \ldots + a_n = 0$ with the condition that $b_0 X^m + \ldots + b_m$ is not zero .

(c) Processing Correlated Equations

In this case, the rule is applied when the highest degree in the equation is even. It is then divided by a term whose degree is $\frac{1}{2}$ of the highest degree in the equation and whose coefficient is 1. For example a bi-quadratic equation $aX^4 + bX^3 + cX^2 + bX + a = 0$ is divided by X^2 to give the correlated equation $a(X^2 + \frac{1}{X^2}) + b(X + \frac{1}{X}) + c = 0$.

(7) Factorization Rule

This is applied in the following two situations. It is applied to $a_0 X^n + \ldots + a_n = 0$ when the left hand side of the equation is a polynomial and the right hand side is zero. Factorization is done using the factorization axiom or the characteristic of an odd degree correlated equation that X+1 is always a factor. It is also applied to equations of the form $(A * B = 0)$ when multiplication is the primary operator of the left hand side and when the right hand side is zero. The equation is factored into two to solve $A = 0$ and $B = 0$. A factored equation that does not contain an unknown variable is excluded.

(8) Expansion Rule

This rule is applied when it is possible to expand the equation. Polynomials, however, are excluded because they are expanded in the process of arriving at a canonical form. There is a danger of falling into an infinite loop when using this rule because it may destroy equations which have already been tidied up by other rules. In order to avoid this situation, the *demo* predicate checks the number of applications of the expansion rule.

4.5 Rewriting Rules for Integration

(1) Integration rule for polynomials

This rule is applied when the input expression is a polynomial. The polynomial normal form consists of a list of elements of the form $[(C_k, N_k), \ldots, (C_0, 0)]$ to represent the polynomial $C_k {}^* X^{N_k} + \ldots + C_0$. When the expression (polynomial) is transformed into its polynomial normal form, the expression can be solved by using the following rule:

$$\int aX^n dX \qquad \rightarrow \qquad \frac{aX^{n+1}}{n+1} + C$$

(2) Integration rule for addition

This rule is applied when the main operator of the input expression is addition. The expression is factored in to two expressions. A typical application of this rule is

$$\int\Big(f(X)+g(X)\Big)dX \qquad \rightarrow \qquad \int f(X)dX + \int g(X)dX$$

(3) Integration rule for expressions containing a constant

This rule is applied when the input expression contains constant factors not unify.

$$\int k*f(X)dX \qquad \rightarrow \qquad k\int f(X)dX$$

k is a constant factor.

(4) Integration rule for basic functions

This rule is applied when the input expression is a basic function. Basic functions are distinguished by the predicate 'is _basic _function'. Some typical application of this rules are

$$\int \sin f(X)dX \qquad \rightarrow \qquad -\frac{1}{f(X)'}*\cos f(X)+C$$

$$\int e^{f(X)}dX \qquad \rightarrow \qquad \frac{1}{f(X)'}*e^{f(X)}+C$$

f(X) is monomial and *f(X)'* is a constant. $\frac{d\,f(X)}{dX}=f(X)'$

(5) Rule of integration by parts

This rule is applied when the input expression has two factors. The standard formula for integration by parts is

$$\int f(X)*g(X)'dX \qquad \rightarrow \qquad f(X)*g(X)+\int f(X)'*g(X)dX$$

This rule is applied for the following human heuristic.

$\int$ *monomial*trigonometric_function dX* $\rightarrow$

apply : Integration by parts
with *f(X) = monomial*,
g(X) = trigonometric _function.

In English, this heuristic means that 'If the current integration is the product of a monomial and a trigonometric function, then try using integration by parts, and bind the operator arguments *f(X)* and *g(X)'* to the indicated factors.'

5. Inference Method

The form and rule controls described so far used meta-level inference. Meta-level inference in AMIE controls object-level inference. Object-level inference expresses rewriting rules such as $A*X + B*X \Rightarrow (A + B)*X$. That is to say meta-level inference is controlled by human heuristic knowledge which can guide the system towards the best way of solving an expression. Figure 5-1 shows a part of the predicate *'strategy'* for selecting rules on the basis of such heuristic knowledge.

Arguments for *'strategy'* represent control information such as input expression, scope of applied rule, control information (information for the next processing, unknowns, and other control data) and various kinds of new control information. *'strategy'* decides the rule to be applied using meta-level knowledge. For example, ST1) to ST5) contain meta-level knowledge: ST2) the isolation rule for equations, ST3,4)factorization rules of equation and ST5) the rule of integration by parts written in them.

ST1) applies rules in accordance with the information for the next processing when such information is input by the user or is stored from past processing.

ST2) decides the application of the isolation rule when the right hand side is free of unknowns (predicate *'free _ of'*), when there is a single occurrence of an unknown in the left hand side (predicate *'singleocc'*), and when its position is known (predicate *'position'*).

ST3) decides the application of a factorization rule when the main operator is multiplication (predicate *'main _ op _ mult'*) and when the right hand side is zero (predicate *'zero'*).

ST4) decides the application of a factorization rule when the left hand side is a polynomial (predicate *'is _ polynomial'*) and the right hand side is zero (predicate *'zero'*).

ST5) decides the application of a rule for integration by parts when the expression consists of two factors (predicate *'number _factor'*) and when, for instance, one function is a monomial and the other is a trigonometric function (predicate *'select _f _dg'*).

6.Determining Degree of Difficulty

By looking at on expression, a human being can tell more or less how difficult the problem is. Such judgment is often made from the complexity and special forms of the equation. A given problem may be solved in a number of ways and the technique applied can make the task easier or more difficult. By adopting three types of rewriting techniques, the degree of difficulty is computed empirically as a cost.

(1) Weighting by ease or difficulty of finding a solution technique.

(2) Weighting by the number of steps involved in solution.

(3) Dynamic weighting by the number of rules in continuous use.

Each weighting process is assumed to be independent and the sum of the three is used as the final degree of difficulty.

(1) Weighting by method of problem solving:

Generally speaking each rewriting rules has different degrees of difficulty. Rewriting $X+B=A \Rightarrow X=A-B$ and $X^2=A \Rightarrow X=\sqrt{A}, X=-\sqrt{A}$, the latter is considered more difficult. The weighting in the latter case will be greater.

(2) Weighting by number of steps:

Again the more steps involved in solution, the greater the volume of computation and hence, the more difficult the problem. In finding factors, the higher the degree the greater the number of substitutions and so the more difficult is the solution. That is so say it is far more difficult to extract factor $X+1$ from $X^4-X^3-7*X^2+X+6$ than from X^2-1.

(3) Weighting by number of rewriting rules:

Generally speaking, the more rules that have to be applied the greater the difficulty. An equation requiring application of a rule is easier to solve than one requiring ten. Therefore a greater cost was attached to equations requiring several rewriting rules.

The weighting process used for calculating the degree of difficulty may be applied to the ordering of CAI teaching material. Efficient selection and application of rules can be achieved by use of this concept in the sequential control of the rules.

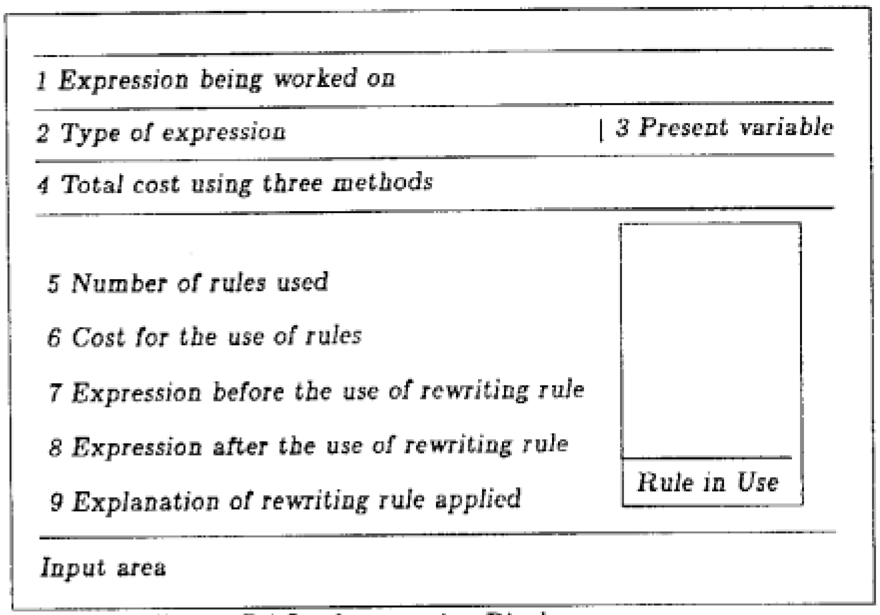
7. Examples

An example of system implementation is shown for Expression 1 from the introduction. Figure 7-1 shows the display panel.

(1) When the equation is input to the system its type is determined by the type control function. The example shown is a polynomial equation and the panel display it as such.

```
ST1) strategy(Eqn,[X],[[Next1|Rest],Var,Ctrl],N_Ctrl) :-
          (strategy(Eqn,[Next1],[ [],Var,Ctrl],N_Ctrl),!, X=Next1 ;
          strategy(Eqn,[World],[Rest,Var,Ctrl],N_Ctrl),!, X=World ).
ST2) strategy(L=R,[isolate],[[],Var,Ctrl],[[],Var,[position(P)|Ctrl]]) :-
          free_of(Var,R),
          singleocc(Var,L),
          position(Var,L,P).
ST3) strategy(L=R,[fact],[[],Var,Ctrl],[[],Var,Ctrl]) :-
          main_op_mult(L),
          zero(R).
ST4) strategy(L=R,[fact],[[],Var,Ctrl],[[],Var,Ctrl]) :-
          is_polynomial(Var,L),
          zero(R).
ST5) strategy(Exp,[int_part],[[],Var,Ctrl],[[],Var,[f_dg(F,DG)|Ctrl]]) :-
          number_factor(Exp,Var,2,FancList),
          select_f_dg(FancList,Var,F,G).
```

Figure 5-1 Part of the predicate 'strategy'


1 Expression being worked on
2 Type of expression
| 3 Present variable
4 Total cost using three methods
5 Number of rules used
6 Cost for the use of rules
7 Expression before the use of rewriting rule
8 Expression after the use of rewriting rule
9 Explanation of rewriting rule applied
Rule in Use
Input area
Figure 7-1 Implementation Display

(Figure 7-2)

(2) Control moves to rule application which decides the rules to be used for successful solution. Equation $f(X) = X^2$ can be rewritten $4*(f(X))^2 - 17*f(X) + 4 = 0$. A substitution rule is used to substitute a new unknown @1 for the unknown X^2, so the equation is now rewritten as $4*@1^2 - 17*@1 + 4 = 0$. (Figure 7-3)

(3) As the equation is quadratic, formula rules are applied so that $@1 = 4$ and $@1 = \frac{1}{4}$. (Figure 7-4)

(4) Now that the equation is separated into two, the system solves for $@1 = 4$. That is, it return @1 to the original form $X^2 = 4$ and solves. (Figure 7-5)

(5) This equation has a single occurrence of an unknown in the left hand side and none in the right hand side. The isolation rule is applied, so that $X = 2, X = -2$. (Figure 7-6)

(6) The system now takes the other part of the equation separated in (4) and solves for $@1 = 1/4$ by returning @1 to the original form $X^2 = \frac{1}{4}$. (Figure 7-7)

(7) The isolation rule is applied as the equation has single occurrence of an unknown in the left hand side and none in the right hand side, so that $x = \frac{1}{2}, X = -\frac{1}{2}$. (Figure 7-8)

(8) With the successful solution of the problem, the system displays the degree of difficulty using method 1 and all solutions. (Figure 7-9)

Results using other solution methods are shown in Figures 7-10 to 7-11. Symbols used in the history of rule applications appear in the order of use. For** represents formula rules, iso**: isolation, col**: collection, att**: attraction, subst**: substitution, replace: returning substitution to original, con**: conversion, fact**: factorization, and expd**: expansion. The history shows that the following solution methods were used in the order of use.

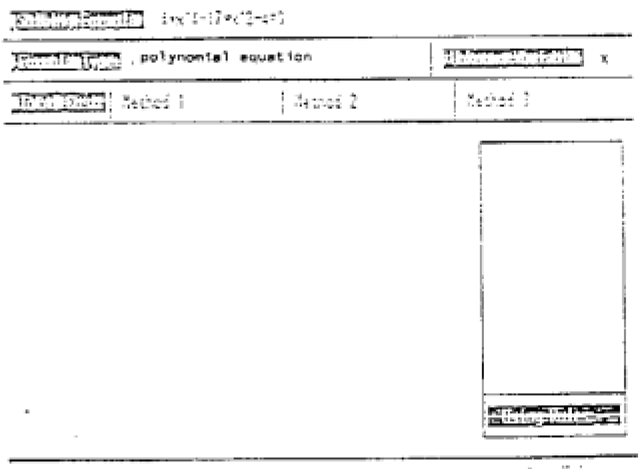
Method 1 solved the equation as a complex quadratic equation,

Method 2 as bi-quadratic correlated equation,

Method 3 by using factorization.

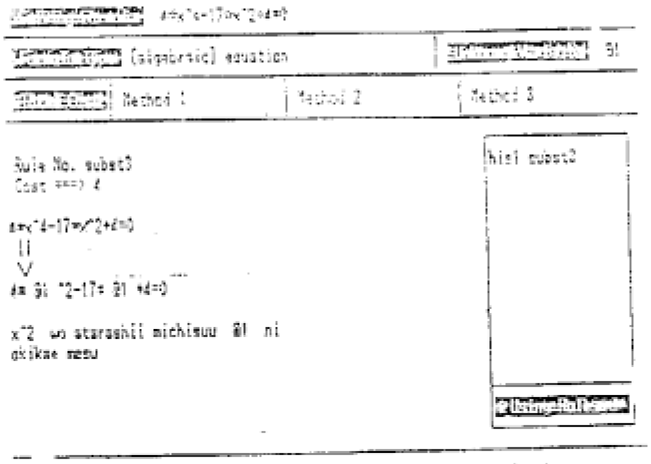
The inference processes for the three methods are more or less the same as that humans use in solving mathematical problems. The degrees of difficulty for each of the methods are given as 16, 36, and 29 respectively, again corresponding roughly to the relative difficulty in solving them based on human empirical knowledge.

The problems which can be solved by AMIE are, as indicated in Figure 1-1, those that appear in high school text books and fundamental university entrance examinations. Some exponential and logarithmic equations for which there are at present no general methods but which depend on numerical and graphic methods must be addressed in the future.



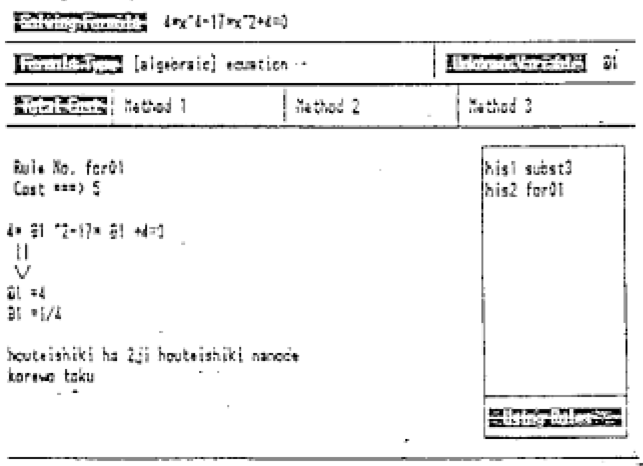
```
4*x^4-17*x^2+4=0
. polynomial equation                    x
Method 1        Method 2        Method 3
```

Figure 7-2 Display Input Expression Type



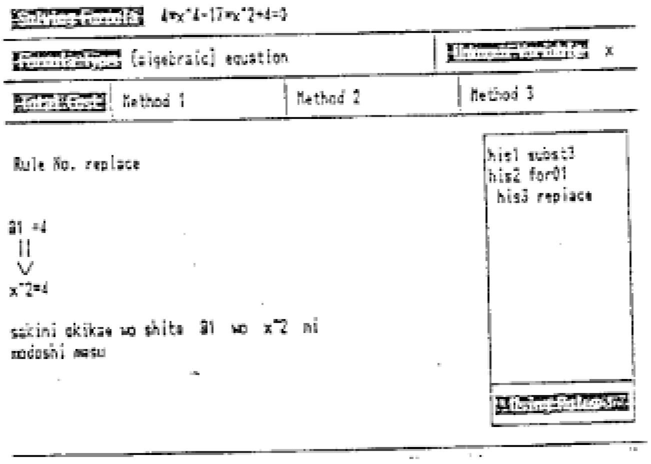
```
4*x^4-17*x^2+4=0
(algebraic) equation                     @1
Method 1        Method 2        Method 3

Rule No. subst3                          his1 subst2
Cost ===> 4

4*x^4-17*x^2+4=0
 ||
 V
4* @1 ^2-17* @1 +4=0

x^2  wo atarashii michisuu  @1  ni
okikae masu
```

Figure 7-3 Substitution Rule Applied



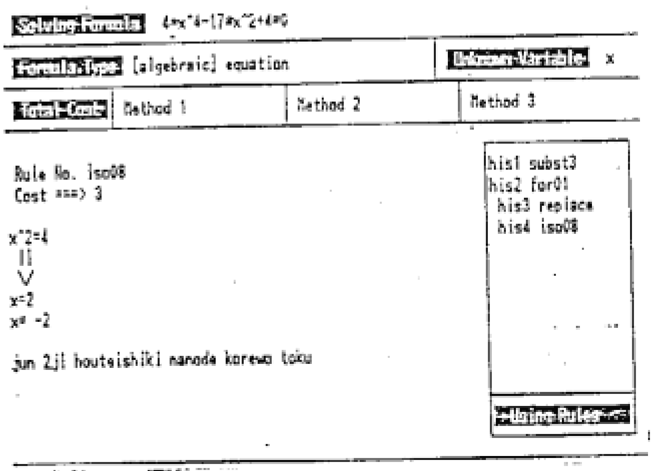
```
4*x^4-17*x^2+4=0
[algebraic] equation                     @1
Method 1        Method 2        Method 3

Rule No. for01                           his1 subst3
Cost ===> 5                              his2 for01

4* @1 ^2-17* @1 +4=0
 ||
 V
@1 =4
@1 =1/4

houteishiki ha 2ji houteishiki nanode
korewo toku
```

Figure 7-4 Formula Rule Applied



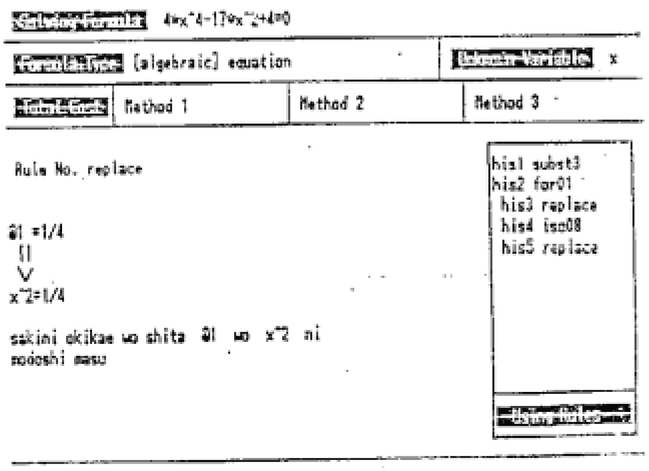
```
4*x^4-17*x^2+4=0
(algebraic) equation                     x
Method 1        Method 2        Method 3

Rule No. replace                         his1 subst3
                                         his2 for01
                                          his3 replace
@1 =4
 ||
 V
x^2=4

sakini okikae wo shita  @1  wo  x^2  ni
modoshi masu
```

Figure 7-5 Variable Replaced



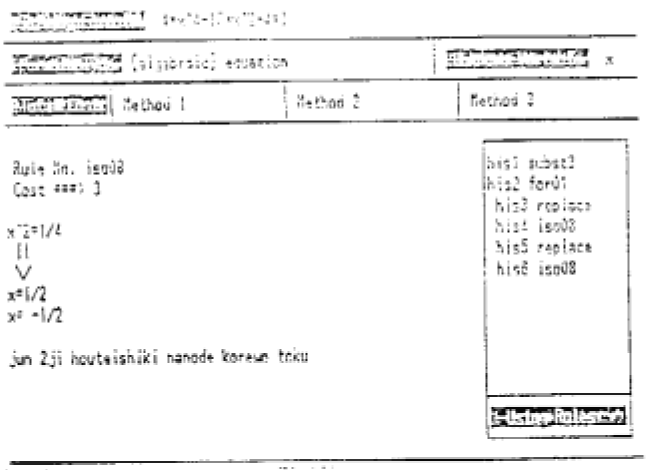
```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1        Method 2        Method 3

Rule No. iso08
Cost ===> 3

x^2=4
 ||
 V
x=2
x= -2

jun 2ji houteishiki nanode korewo toku

his1 subst3
his2 for01
 his3 replace
 his4 iso08
```

Figure 7-6 Isolation Rule Applied



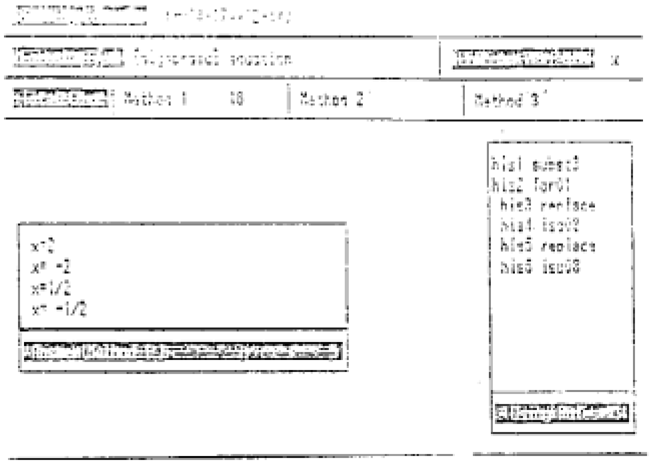
```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1        Method 2        Method 3

Rule No. replace

@1 =1/4
 ||
 V
x^2=1/4

sakini okikae wo shita  @1  wo  x^2  ni
modoshi masu

his1 subst3
his2 for01
 his3 replace
 his4 iso08
 his5 replace
```

Figure 7-7 Variable Replaced



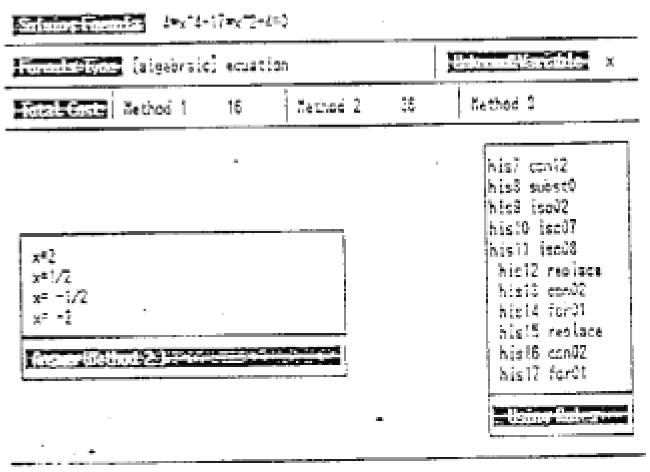
```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1        Method 2        Method 3

Rule No. iso08
Cost ===> 3

x^2=1/4
 ||
 V
x=1/2
x= -1/2

jun 2ji houteishiki nanode korewo toku

his1 subst3
his2 for01
 his3 replace
 his4 iso08
 his5 replace
 his6 iso08
```

Figure 7-8 Isolation Rule Applied



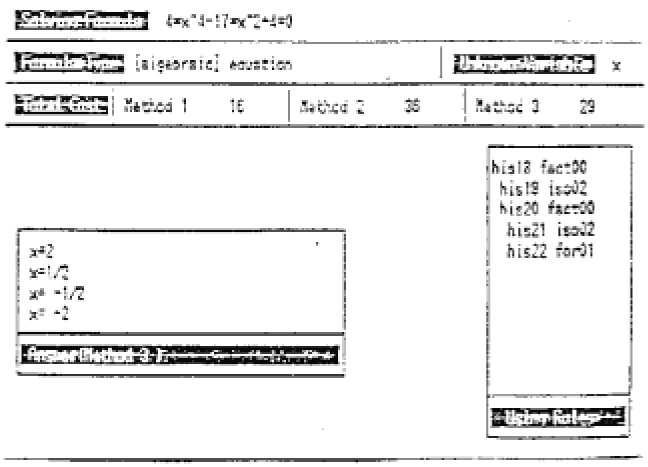
```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1     16     Method 2        Method 3

x=2
x= -2
x=1/2
x= -1/2

his1 subst3
his2 for01
 his3 replace
 his4 iso08
 his5 replace
 his6 iso08
```

Figure 7-9 Display Answers Using Method 1



```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1     16     Method 2     36     Method 3

x=2
x=1/2
x= -1/2
x= -2

his7 con12
his8 subst0
his9 iso02
his10 iso07
his11 iso08
 his12 replace
 his13 con02
 his14 for01
 his15 replace
 his16 con02
 his17 for01
```

Figure 7-10 Display Answers Using Method 2



```
4*x^4-17*x^2+4=0
[algebraic] equation                x
Method 1     16     Method 2     36     Method 3     29

x=2
x=1/2
x= -1/2
x= -2

his18 fact00
 his19 iso02
 his20 fact00
  his21 iso02
  his22 for01
```

Figure 7-11 Display Answers Using Method 3

8. Conclusion

AMIE executes both selection and application of various kinds of rewriting rules and substitution functions using meta-inference according to human heuristics. This method of AMIE builds effective problem solving functions for algebraic problems because of the good heuristic strategies used in meta-inference. Clear expression which discriminates meta-level knowledge and object level knowledge for meta inference makes it easy to add and update both level knowledge. Therefore, users can easily expand an object problem area.

Rewriting rules are categorized according to some characteristic in AMIE. The cost of problem solving is defined by not only an applied method but also the category of the method. Therefore, definition of kinds of categories to which new rules belong, should be evaluated precisely when new problem area is added into the system. Explanation functions of AMIE show the degree of necessity of applied rules and transformation methods. Multiple windows is used to the process of problem solving clearly. Rough degree of difficulty for each solving method in humans experiment is shown by the relative degree of difficulty displayed corresponding to an applied method.

ACKNOWLEDGMENTS

The authors wish to express their thanks to Kazuhiro Fuchi, Director of the ICOT Research Center and to the members of the Second Research Laboratory at ICOT for their valuable comments.

REFERENCES

[Bundy 81] Bundy, A. and Welham, B., "Using Meta-level Inference for Selective Application of Multiple Rewrite Rule Sets in Algebraic Manipulation," Artificial Intelligence No.16, pp.189-210 (1981).

[Bowen 81] Bowen, A. and Kowalski, R., "Amalgamating Language and Meta-language in Logic Programming," June (1981).

[Furukawa 84] Furukawa, K., Kunifuji, S., Takeuchi, A., and Ueda, K., "The Conceptual Specification of the Kernel Language Version 1", ICOT TR-054, (1984).

[Hearn 84] Hearn, A. C., "REDUCE User's Manual, Version 3.1, The Rand Corp. CP-78, Santa Monica (1984).

[Jenks 84] Jenks, R. D., "A Primer : 11 Keys to New SCRATCHPAD, Lecture Notes in Computer Science, No.174, pp.123-147, Springer-Verlag, Berlin (1984).

[Mathlab 77] The Mathlab Group, " MACSYMA Reference Manual, MIT, Massachusetts (1977).

[Miyachi 84] Miyachi, T., Kunifuji, S., Kitakami, H., Furukawa, K., Takeuchi, A., and Yokota, H., "A Knowledge Assimilation Method for Logic Databases," IEEE 1984 International Symposium on Logic Programming , pp.118-125 (1984).

[Slagle 61] Slagle, J. R., "A Heuristic Program the Solves Symbolic Integration Problem in Freshman Calculus, Ph,D. Dissertation, Harvard Univ., Cambridge, Mass. (1961).

[Sterling 82] Sterling, L., Bundy, A., Byrd, L., O'Keefe, R., and Silver, B., "Solving Symbolic Equations with PRESS," DAI Research Paper 171, Dept. of A.I. Univ. of Edinburgh (1982)