A Study of Computer-Based Problem Solving Based on Equivalent Transformations

Yuuichi KAWAGUCHI
yuuichi@tenshi. ac. jp

In order to solve a given problem by using a computer, we need an algorithm. If an algorithm is produced with being based on human inspiration, then it is difficult to guarantee the corectness and efficiency of the algorithm. Correctness and efficiency are important sides of quality of algorithms. It is necessary that algorithms are produced with being based on theories.

A given problem is expressed by using a formal language. Algorithms consists of means to operate such expressed problems and means to control applications of operations. In this study, I only use equivalent transformations for such operations. Equivalent transformations guarantee the correctness of algorithms. Appropriate control of applying equivalent transformations makes algorithms efficient.

Key words : Computer-Based Problem Solving
Equivalent Transformation
Correctness of Algorithm
Efficiency of Algorithm

天使大学 看護栄養学部 教養教育科

1 Introduction

Correctness and efficiency are important sides of quality of algorithms. If an algorithm is produced with being based on human inspiration, then it is difficult to guarantee both the correctness and the efficiency. It is necessary that algorithms are produced with being based on theories, not human inspiration.

This note explains a programming paradigm. In the paradigm, any equivalent transformations are allowed as means of computing. The correctness of algorithms is guaranteed. Since means of computing are selected from many equivalent transformations, it is highly possible that an efficient algorithm is produced. This note explains how to guarantee the correctness of an algorithm and why an efficient algorithm can be produced.

2 Production of Algorithm

If a method to produce algorithms is based on human inspiration, then there is a problem. Human inspiration is unstable. One day a person may produce a high quality algorithm, but in another day the same person may produce a low quality one. In the case, it is difficult to keep a quality of the produced algorithm at a constant level.

In order to use algorithms as general and stable tools for solving problems, they are produced with a constant quality similarly to industrial products. Algorithms must be produced with being based on theories, not human inspiration. An algorithm that solves a given problem efficiently may be produced by a programmer with a great ability. However, similarly to music or fine arts, if an algorithm is based on an unstable human inspiration, it is difficult to use it as a general tool for solving problems.

Here I noticed the efficiency of algorithms as one side of a quality of them. There is one more important side of a quality. That is a correctness. When an algorithm is produced with being based on human inspiration, it is difficult to prove that the algorithm is correct. Let us consider, for example, solving an equation. There are some elementary operations, such as adding a same value to each side, which were proven as correct. If an algorithm consist of only such elementary operations, then it is easy to prove that the algorithm is correct. For other cases where there are no elementary operation proven as correct, it is difficult to prove either theoretically or intuitively that an algorithm is correct.

There are two essential points:

(1) The quality of an algorithm consists of an efficiency and a correctness.
(2) If an algorithm is produced with being based on human inspiration, then it is difficult to guarantee its quality.

These points result in the following conclusion; algorithms must be produced automatically with being based on theories.

3 Expression of Problem

When a problem is given, there is a case where the problem is stated in a natural language. The statement may include ambiguities. In another case, the problem exists only in one's brain vaguely. In order to solve the problem correctly, it is needed to denote it without any ambiguity. When an ambiguous problem is given, it is impossible to decide if a method for solving a problem and a produced answer are correct or not. We need a formal description of a given problem. It is called the "expression" of the problem.

A method for solving a problem, *ie.*, an algorithm, can be produced only if the expression of it is given. Thus, the expression of the problem is denoted before an algorithm for solving it being produced. The expression is independent of the algorithm.

4 Correctness

Let an expression of a problem be d, and the answer of it be a. A map f combines d with a. Formally, I assume that $a=f(d)$ holds. This note covers only an area where the formula holds. Let a domain of f be D, and a region of f be R.

The map f is defined against any element in D. An element in D is an expression of some problem. Depending on an expression $d \in D$, it may be difficult or may take a long time to compute $f(d)$. In such cases, we should not compute $f(d)$ directly.

Here I assume the following two:

(1) For $D' \subset D$, a map g: $D' \rightarrow R$ exists and for $d' \in D'$, $f(d')=g(d')$ holds.

(2) For $d' \in D'$, it is easier to compute $g(d')$ than to compute $f(d')$.

For any expression $d' \in D'$, it is reasonable to use the map g rather than f for computing the answer of the given problem.

To summarize, when an expression $d \in D$ of a problem is given, a method for solving is:

1. For $d \in D$, produce $d' \in D'$ by applying a transformation t to d.
2. We have the answer a of the transformed expression d' by computing $a=g(d')$.

If $a=f(d)=f(d')=g(d')$ holds, then this method is correct. From the assumptions $a=f(d)$ and $f(d')=g(d')$ always hold. Thus, the method for solving described above is correct, if $f(d)=f(d')$ holds. This means that $a \in R$ is kept unchanged after applying the transformation t to $d \in D$ and producin $d' \in D'$.

When a transformation t transforms d to d' and satisfies $f(d)=f(d')$, then it is called an "equivalent transformation."

If we use only equivalent transformations, then the method for producing d' from d is correct. In general, it is difficult to produce $d' \in D'$ from $d \in D$ by applying an equivalent transformation at only one time. Computation then is a repetitious applications of equivalent transformations to the expression d.

In order to solve a given problem, the problem is expressed by an expression $d \in D$ and then $d' \in D'$ is produced by applying some equivalent transformations $t_1, \ldots\ldots, t_n$ to d, *ie.*,

$$d \in D \xrightarrow{t_1} d_1 \xrightarrow{t_2} d_2 \ldots \xrightarrow{t_n} d_n = d' \in D'.$$

If $d' \in D$ is produced, then it is possible to produce the answer a by $a=g(d')$.

Since all transformations $t_i\,(i=1,\ldots,n)$ are the equivalent transformation,

$$\begin{aligned} a &= f(d)=f(d_1)=f(d_2) \\ &= \cdots \\ &= f(d_n)=f(d')=g(d') \end{aligned}$$

is guaranteed. This formula guarantees the correctness of this method for solving the problem.

5 The ET Paradigm

I call this approach for solving problems the "equivalent transformation paradigm," or the ET paradigm for the abbreviation. The ET paradigm does not offer any concrete system for expressing. It only offers that a map f combines between an expression of a problem d and the answer a of it.

The ET paradigm assumes that there are a map $f: D \rightarrow R$, subset $D' \subset D$ and a map g: $D' \rightarrow R$ exist, and assumes that for an expression $d' \in D'$ it is easier to compute $g(d')$ than to compute $f(d')$.

The execution in the ET paradigm is a production of $d' \in D'$ from the original $d \in D$ by applying equivalent transformation repeatedly to the original d. This paradigm loosely combines the expression of a given problem and an execution.

For example, in the case of the Logic Programming (LP) paradigm [1, 2], a problem is expressed by using definite clauses and an execution is based on the SLD derivation. The SLD derivation is only free to decide an order of selecting a clause and an atomic formula. Thus, in the LP paradigm, an expression of a problem and an execution are tightly combined. Even if there is a better method for executing than a method not based on the SLD derivation, it can not be used. The LP paradigm guarantees the correctness of an execution, *ie.*, algorithm, if it is based on SLD paradigm. The area to be guaranteed as correct is, however, narrow.

As compared with the LP paradigm, the ET paradigm loosely combines an expression with an execution. The ET paradigm guarantees that the execution, *ie.*, algorithm, is correct, if we use only but any equivalent transformations. The area to be guaranteed is wide. Therefore, there is a high possibility to find a better method for executing.

6 Example

6.1 Simultaneous Linear Equations

Let us consider solving a problem of simultanious linear equations:

$$\begin{cases} a_{11}x + a_{12}y = c_1 \\ a_{21}x + a_{22}y = c_2 \end{cases} \qquad (1)$$

Symbols $a_{11}, \ldots, a_{22}$, c_1 and c_2 are real numbers, and x and y are unknown. A problem is to be find real numbers x and y satisfying (1). This problem is solved by using matrixes. Let a matrix A be $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, a matrix X be $\begin{pmatrix} x \\ y \end{pmatrix}$, and a matrix C be $\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$.

Suppose the inverse matrix of A exists and is denoted as A^{-1}. The simultanious linear equations (1) is denoted as $AX = C$. By multiplying A^{-1} to both sides from the left, we can produce the answer.

$$A^{-1}AX = A^{-1}C$$
$$X = A^{-1}C$$

There are three elementary transformations for matrixes.

$E_i(t)$: multiply all elements in the ith column by t.
$E_{ij}(t)$: add the ith column multiplied by t to the jth column.
P_{ij}: exchange ith and jth column.

If A is regular, mathematical theories guarantee that there is a sequence $T_1, \ldots, T_n$ $(n \neq \infty)$ satisfying $T_1 \cdots T_n \cdot A = A^{-1}$ where for $k = 1, \ldots, n$ each T_k is one of $E_i(t)$, $E_{ij}(t)$ and P_{ij}. Each T_k $(k = 1, \ldots, n)$ has some parameters. They depend on A. Here, I can not show the concrete value of them, since A is abstract.

According to the ET paradigm, let us formalize this situation. The problem is given with a simultanious linear equations. Since these equations are characterized by two matrixes A and C, I denote the expression d of the problem as $\langle A, C\rangle$. The answer a is a 2×1 matrix satisfying $Aa = C$. The map f is defined as f: $\langle P, Q\rangle \mapsto S$ satisfying $PS = Q$, where P is any 2×2 matrix, and S and Q is any 2×1 matrix. The domain D of f is a set of all such pairs $\langle P, Q\rangle$. The range R of f is a set of all 2×1 matrix. If P is not regular, then the matrix $f(\langle P, Q\rangle)$ is undefined.

Here we have the equation $f(d) = a$, which is required by the ET paradigm.

Let us consider a set D' of all pairs $\langle E, Q\rangle$ where E is an unit matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and Q is any 2×1 matrix. The set D' is a subset of D. An expression $d' = \langle E, C'\rangle \in D'$ represents an equation $EX = C'$, *ie.*, $X = C'$, where X is $\begin{pmatrix} x \\ y \end{pmatrix}$. Thus, the answer of d' is C'. The map g: $D' \to R$ is defined as g: $\langle E, Q\rangle \mapsto Q$. This is a trivial map.

Here we have f: $D \to R$ and g: $D' \to R$, where the relation $D \subset D'$ is satisfied and for $d' \in D'$ and $d \in D$ it is easier to compute $g(d')$ than to compute $f(d)$.

There is another component demanded by the ET paradigm: a set T of equivalent transformations. By applying each element of T to $d \in D$ in some times, the original problem d is equivalently transformed into $d' \in D'$.

Let T be a set of the three elementary transformations, $E_i(t)$, $E_{ij}(t)$ and P_{ij}. If a original problem d is $d = \langle A, C\rangle$ and $t \in T$ is applied to d, then we have new problem $d_1 = \langle tA, tC\rangle$. In this case $f(d) = f(d_1)$ holds. I omit the proof. Therefore, each element of T is an equivalent transformation.

Now we have all requirements of the ET paradigm.

It is not trivial to produce the sequence $T_1, \ldots, T_n \in T$ satisfying $T_1 \cdots T_n \cdot A = A^{-1}$. There are some methods for computing the sequence, such as Gauss–Jordan's method. The method is not only one. The ET paradigm allows any other method if it only uses equivalent transformations. There may be a more efficient method than Gauss–Jordan's. If so, we sholud use that.

6.2 Discussion

In Section 6.1, according to the ET paradigm, I formalized the problem of solving simultaneous linear equations. The original problem $d = \langle A, C\rangle$ is transformed into $d' = \langle E, C'\rangle$ by applying some equivalent transformations T_k $(k = 1, \ldots, n)$, and then the answer of the original problem is C'. This computation is correct. The components required by the ET paradigm is a set $\langle f: D \to R,\ g: D' \to R,\ T\rangle$ that satisfying the condition. It explains the situation completely.

I hope that the ET paradigm can explain many other situations. I am researching other examples.

7 Past and Present Work

I published papers about the following subjects:

1. a formal language for expressing problems [3],
2. type inferences [4], and
3. higher order computations [5].

The ET paradigm does not offer any system for expressing problems. The example shown above uses mathematics for it. I used the predicate logic that treats directly class hierarchies and substructures. Those papers begin with constructing the system.

These resarches are examples of concrete applications of the ET paradigm, and devote to examine an area covered by the ET paradigm. If there are more examples, the area is wider. I am also interested in the abstract side of the ET paradigm. A set $\alpha = \langle f: D \to R, g: D' \to R, T\rangle$ that satisfies the conditions characterizes an area where problems are treated. When there are two sets α_1 and α_2, it is interesting to examine which set covers the wider area, or to examine if the two sets are homomorphic or not. These resarches devote to examine whether any other component is necessary for the ET paradigm to characterize the area completely.

8 Conclusion

This note explained the ET paradigm.

As to the quality, I mainly explained about the correctness of algorithm. There was another side of the quality, *ie.*, efficiency. I aim at producing efficient algorithms automatically. I, however, have no prominent result as to the efficiency.

The ET paradigm is hoped to cover a wide area of computations, since the ET paradigm loosely combines a problem with the answer of it. I think that the looseness causes to find an efficient algorithm.

Acknowledgments

The idea of the ET paradigm originated from Professor Akama K. I am organizing it in my style. He helped me to publish my doctoral thesis [6]. Discussions with Dr. Ogurisu O. about the ET paradigm was worthwhile. The anonymous reviewer gave me suggestive advises. I thank all of them.

References

[1] Robert A. Kowalski. *Logic for Problem Solving*. Elsevier North Holland, Inc., 1979. translated into Japanese.

[2] John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag, second edition, 1987.

[3] Yuuichi Kawaguchi, Kiyoshi Akama, and Eiichi Miyamoto. Representation and calculation of objects with classes and substructures—a simple computational framework based on logic—. *Journal of Jsai.*, 12(1): 48-57, 1997. in Japanese.

[4] Yuuichi Kawaguchi, Kiyoshi Akama, and Eiichi Miyamoto. Applying program transformation to type inference on a logic language. *IEICE Trans. on Inf. & Syst.*, E81-D(11):1141-1147, November 1998.

[5] Yuuichi Kawaguchi, Kiyoshi Akama, and Eiichi Miyamoto. Two basic objects for higher-order expression. In M. H. Hamza, editor, *Proceedings of the ISATED International Conference: Applied Informatics (AI'2000), pages* 407-410. IASTED, February 14-17 2000.

[6] Yuuichi Kawaguchi. *A Study on Formalizing Expression and Execution Of Objects that Have Class Hierarchy and Substructure.* PhD Thesis, Hokkaido University, May 2000. in Japanese.