記号処理 2 － 4
(1977. 10. 6)

AN AUTOMATIC THEOREM PROVER GENERATING A PROOF IN NATURAL LANGUAGE

Masakazu NAKANISHI, Morio NAGATA and Kenji UEDA
Faculty of Engeneering, KEIO University
Yokohama, JAPAN

ABSTRACT

An automatic verifier, called TKP 2, to prove correctness of recursive programs is presented. It generates an effective proof in English. The implementation method of the theorem prover is shown with an example.

1. INTRODUCTION

Verification of programs is a very important problem in software engineering. In order to prove correctness of programs, a number of automatic verifiers have been designed [1],[5]. But most of them display proofs with their own peculiar forms, then it tends to be difficult to understand their proofs even for logicians or engineers.

We implemented a theorem prover called TKP 1.2 to prove correctness of recursive programs in 1975 [2],[4]. Though it can prove various properties of recursive programs and the proof figure generated by this system has the form to be easily read for a man, it may not be familiar with some mathematicians. Moreover the proof figure contains many steps which seem to be trivial and redandunt for mathematicians.

We have designed and implemented a new version of TKP 1.2 which is called TKP 2. It generates an effective proof in English as if it had proved correctness in human-like way. This paper describes the implementation method of this new system with an example.

2. VERIFICATION OF RECURSIVE PROGRAMS

2.1 NOTATIONS

The methodology of our system is based on the fixed point approach of mathematical theory of computation and a Gentzen-type formal system representing properties of functions by T.Nishimura [3]. We introduce notations used in this formal system.

Now let us consider two Lisp programs, f[x] and g[x;y], computing factorials of integers.

```
f[x]=[zerop[x]→1;
      t→x×f[x-1]]

g[x;y]=[zerop[x]→y;
      t→g[x-1;x×y]]
```

The least fixed point of f with a formal parameter x is written as

$$\bigvee_{n=0}^{\infty} f^n(x),$$

where $f^n(x)$ is denoted as

$$f^n(x) = p(x) \wedge 1 \vee \neg p(x) \wedge x \times f^{n-1}(x-1) \qquad (1)$$

$f^0(x)$ is a totally undefined function and its value is undefined. n designates the number of the applications of functions.

$g^n(x,y)$ is expressed as

$$g^n(x,y) = p(x) \wedge y \vee \neg p(x) \wedge g^{n-1}(x-1, x \times y) \qquad (2)$$

2.2 PROPERTY OF RECURSIVE PROGRAMS

Properties of the fixed point of a recursive program are considered. For example, a property of the two recursive programsm may be

if $\alpha = f^0(k) \vee \alpha = f^1(k) \vee \alpha = f^2(k) \vee \cdots \vee \alpha = f^n(k) \vee \cdots$
then $\alpha = g^0(k,1) \vee \alpha = g^1(k,1) \vee \alpha = g^2(k,1) \vee \cdots \vee \alpha = g^m(k,1) \vee \cdots$.

We abbreviate above statement as follows.

$$\text{if } \alpha = \bigvee_{n=0} f(k) \quad \text{then } \alpha = \bigvee_{m=0} g(k,1) \qquad (3)$$

In order to illustrate the proving process of TKP 2, we will describe the proof of the property (3).

2.3 Mathematical Induction

TKP 2 proves a property of a recursive program by mathematical induction on the number of applications of functions. Namely, the property is proved in induction basis and induction step. The given property will be proved for $n=0$ at first. In this case it is written as

$$\text{if } a=f^{n}(k) \quad \text{then } a=\bigvee_{m=0}^{\infty} g^{m}(k,1).$$

Since $f^{n}(k)$ is undefined, $a=f^{n}(k)$ is false, and the above statement holds. Next the hypothesis of the induction is given such that the following statement is valid for every $\ell < n+1$.

$$\text{if } a=f^{\ell}(k) \quad \text{then } a=g^{\ell}(k,1).$$

TKP 2 generates automatically the hypothesis of the induction from the given program.

In this case, the program is to proved that

$$\text{if } a=f^{n+1}(k) \quad \text{then} \quad a=\bigvee_{m=0}^{\infty} g^{m}(k,1). \tag{4}$$

In order to prove (4), it is sufficient to prove that

$$\text{if } a=f^{n+1}(k) \quad \text{then } a=g^{n+1}(k,1). \tag{5}$$

2.4 User Supplied Knowledge

Definitions of F and G are supplied by user in forms of (1) and (2). According to these definitions, TKP 2 proves (5). By definition (1), the statement (5) is equivalent to

$$\text{if } a=p(k)\wedge 1 \vee \neg p(k)\wedge x\times f^{n}(k-1) \quad \text{then } a=g^{n+1}(k,1). \tag{6}$$

There are two kinds of user supplied nowledge. One of them is a definition. Definitions represents programs. The other is a theorem. Theorems are assumed that are already proved. An example of the theorem used in this problem is

$$x \times g^{n}(y,z) = g^{n}(y,x\times z). \tag{7}$$

2.5 Use of Built-in Properties

Let us continue the proof of the statement (6). TKP 2 attempts to prove in two cases. p(k) is assumed true in one case. p(k) is false in the other case.

If we assume that p(k) is true, the statement (6) is transformed into

$$\text{if } a=1 \quad \text{then } a=g^{n+1}(k,1). \tag{8}$$

This is valid by the following process.

Here,

$$\text{if } a=1 \quad \text{then } a=1$$

is trivial. when p(k) is true, $g^{n}(k,1)$ is 1 by definition (2). Therefore, (8) is valid. The property is proved in one case.

Now, let us consider the proof in the other case. If p(k) is assumed false, the statement (6) is equivalent to the following statement.

$$\text{if } a=k\times f^{n}(k-1) \quad \text{then } a=g^{n+1}(k,1). \tag{9}$$

By the hypothesis of the induction, this is transformed into the following.

$$\text{if } a=k\times g^{n}(k-1,1) \quad \text{then } a=g^{n+1}(k,1). \tag{10}$$

Here,

$$\text{if } a=g^{n}(k-1,k) \quad \text{then } a=g^{n}(k-1,k)$$

is trivial.

By theorem (7), this is equivalent to that,

$$\text{if } a=k\times g^{n}(k-1,1) \quad \text{then } a=g^{n}(k-1,k)$$

Because p(k) is false, this is the same statement of (10). Therefore, (9) is proved. Then, this completes the proof.

2.6 Visualizer of TKP 2

The proof of TKP 2 is almost same procedure described in previous sections.

This system is written in the Lisp language on PDP 11 with 28K words. There are three phases in TKP 2. *Translator* translates expression in the mathematical notation into prefix notation. *Prover* performs a proof. *Visualizer* writes the proof in natural language.

The proof tree generated by the prover has many nodes to traverse lofical steps. Main purpose of the visualizer of TKP 2 is to reduce some steps generated by the prover which seems to be trivial or redundant.

We show the algolithm of visualizer.

ALGORITHM

1. If mathematical induction is applied, display the proof of induction basis and the proof of induction step.
2. Eliminate inherently true formulae in the antecedent and inherently false formulae in the consequent. So the undefined formulae placed in the consequent are eliminated.

3. Assign true or false to predicates in formulae , interpret them, and display the interpretation massages to the effect that the value is assigned. As the result, all predicates are eliminated from formulae.
4. In principle, the output is formed in top-to-bottom order. But at the node that has only one leaf to be arrived, it is formed in bottom-to-top order(Fig. 1).

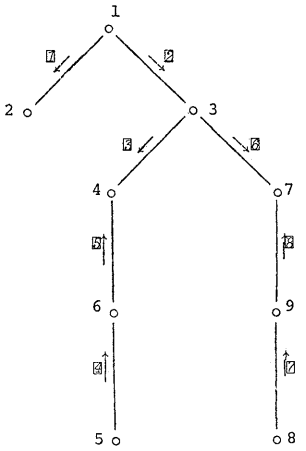


Fig. 1 The order of the Output.

2.7 Experiments

TKP 2 tried to prove same problems which were proved by TKP 1.2.

Problem 1 is a property on factorial programs stated in the previous sections. Problem 2 and 3 are the following.

Problem 2

PROVE THAT

IF A = G(?F(X,Y),Z) THEN A = ?F(X,G(Y,Z))

WHERE

1. F<N+1>(X,Y) = P(X)&YV\P(X)&C(H(X),F<N>(T(X),Y))

USING A THEOREM

1. IF A = G(C(H(X),Y),Z) THEN A = C(H(X),G(Y,Z))

Problem 3

PROVE THAT

IF A = ?F(X,Y) THEN A = (X+Y)/2

WHERE

1. F<N+1>(X,Y) = P(X,Y)&(X+Y)/2V\P(X,Y)&F<N>(F<N>(X-2,Y+1),F<N>(X-1,Y+2))

USING THEOREMS

1. IF A = X/2+Y/2 THEN A = (X+Y)/2
2. IF A = X-2+Y+1 THEN A = (X+Y)-1
3. IF A = X-1+Y+2 THEN A = X+Y+1
4. IF A = Z-1+Z+1 THEN A = Z+Z
5. IF A = Z+Z THEN A = 2*Z
6. IF A = (2*Y)/2 THEN A = Y

The comparison of proving steps is shown in Table 1.

Table 1.

	Problem 1	Problem 2	Problem 3
TKP 2	12	12	13
TKP 1.2	29	33	20

The output of TKP 2 about Problem 1 will be shown in Appendix A. The proof of TKP 1.2 will be in Appendix B.

3. Conclusion

We have attempted to make a mechanical verifier of recursive programs which proves property of programs and writes the proof in human-like way. The proof is efficient and easy to understand for a man. We hope that this prover becomes to be effective to verify or develop recursive programs.

Acknowledgement. The authors would like to thank Prof. Hidetoshi Takahashi of Keio University and Prof. Toshio Nishimura of the University of Tsukuba for their valuable comments.

References

[1] Boyer, R.S. and Moore, J.S. "Proving theorems about LISP functions",*JACM*, Vol.22, No.1, 1975, pp. 129-144

[2] Nakanishi, M., Nagata, M. and Nishimura, T. "TKP 1.2 - The extension of TKP 1 by adding some facilities", *Science Rep. of Tokyo Kyoiku daigaku*, Section A, Vol.13, No.357, 1975, pp. 82-89

[3] Nishimura, T. "Gentzen-type formal system representing properties of functions". *Comment. Math. Univ. St. Pauli*, 23-1,1974, pp. 37-44

[4] Nishimura, T., Nakanishi, M., Nagata, M. and Iwamaru, Y. "Gentzen-type formal system representing properties of functions and its implementation", *Proc. of 4th Int. Joint Conf. of Artificial Intelligence*, Tbilisi, Sept. 1975, pp. 57-64

[5] Suzuki, N. "Verifying programs by algebraic and logical reduction", *Proc, of Int. Conf. Reliable Software*, Los Angels, Apr. 1975, pp. 473-481

APPENDIX A

PROVE THAT

IF A = ?F(K) THEN A = ?G(K,1)

WHERE

1. F<N+1>(X) = P(X)&1V\P(X)&X*F<N>(X-1)

2. G<N+1>(X,Y) = P(X)&YV\P(X)&G<N>(X-1,X*Y)

USING A THEOREM

1. IF A = X*G<N>(Y,Z) THEN A = G<N>(Y,X*Z)

PROOF

IF A = ?F(K) THEN A = ?G(K,1)

WILL BE PROVED BY INDUCTION ON N.

BASIS. FOR N = 0

[1] IF A = F<0>(K) THEN A = ?G(K,1).

THIS IS VALID, BECAUSE F<0>(K) IS UNDEFINED.

INDUCTION STEP. ASSUME (AS HYPOTHESIS OF THE INDUCTION) THAT, FOR EVERY L < N+1,

IF A = F<L>(K) THEN A = G<L>(K,1).

NOW LET'S PROVE

[2] IF A = F<N+1>(K) THEN A = ?G(K,1).

IT IS SUFFICIENT TO PROVE THAT

[3] IF A = F<N+1>(K) THEN A = G<N+1>(K,1).

BY DEFINITION 1, [3] IS EQUIVALENT TO

[4] IF A = P(K)&1V\P(K)&K*F<N>(K-1) THEN A = G<N+1>(K,1).

WE DISTINGUISH TWO CASES.

CASE (1): ASSUME THAT P(K) IS TRUE. THE FOLLOWING STATEMENT IS OBTAINED.

[5] IF A = 1 THEN A = G<N+1>(K,1).

NOW, THE FOLLOWING STATEMENT IS VALID.

[6] IF A = 1 THEN A = 1.

SINCE P(K) IS TRUE, [6] MAY BE WRITTEN AS FOLLOWS.

[7] IF A = 1 THEN A = P(K)&1V\P(K)&G<N>(K-1,K*1).

BY DEFINITION 2,

IF A = 1 THEN A = G<N+1>(K,1).

CASE (1) IS PROVED.

CASE (2): ASSUME THAT P(K) IS FALSE. THE FOLLOWING STATEMENT IS OBTAINED.

[8] IF A = K*F<N>(K-1) THEN A = G<N+1>(K,1).

BY THE HYPOTHESIS OF THE INDUCTION, IT IS SUFFICIENT TO PROVE THAT

[9] IF A = K*G<N>(K-1,1) THEN A = G<N+1>(K,1).

NOW, CONSIDER THE FOLLOWING VALID STATEMENT.

[10] IF A = G<N>(K-1,K*1) THEN A = G<N>(K-1,K*1).

SINCE P(K) IS FALSE, [10] MAY BE WRITTEN AS FOLLOWS.

[11] IF A = G<N>(K-1,K*1) THEN A = P(K)&1V\P(K)&G<N>(K-1,K*1).

BY DEFINITION 2,

[12] IF A = G<N>(K-1,K*1) THEN A = G<N+1>(K,1).

BY THEOREM 1,

IF A = K*G<N>(K-1,1) THEN A = G<N+1>(K,1).

CASE (2) IS PROVED.

THIS COMPLETES THE PROOF.

Appendix B

PROVE ?F(K) -> ?G(K,1)

WITH DEFINITIONS

1. F<N+1>(X) = P(X)&1V\P(X)&X*F<N>(X-1)
2. G<N+1>(X,Y) = P(X)&YV\P(X)&G<N>(X-1,X*Y)

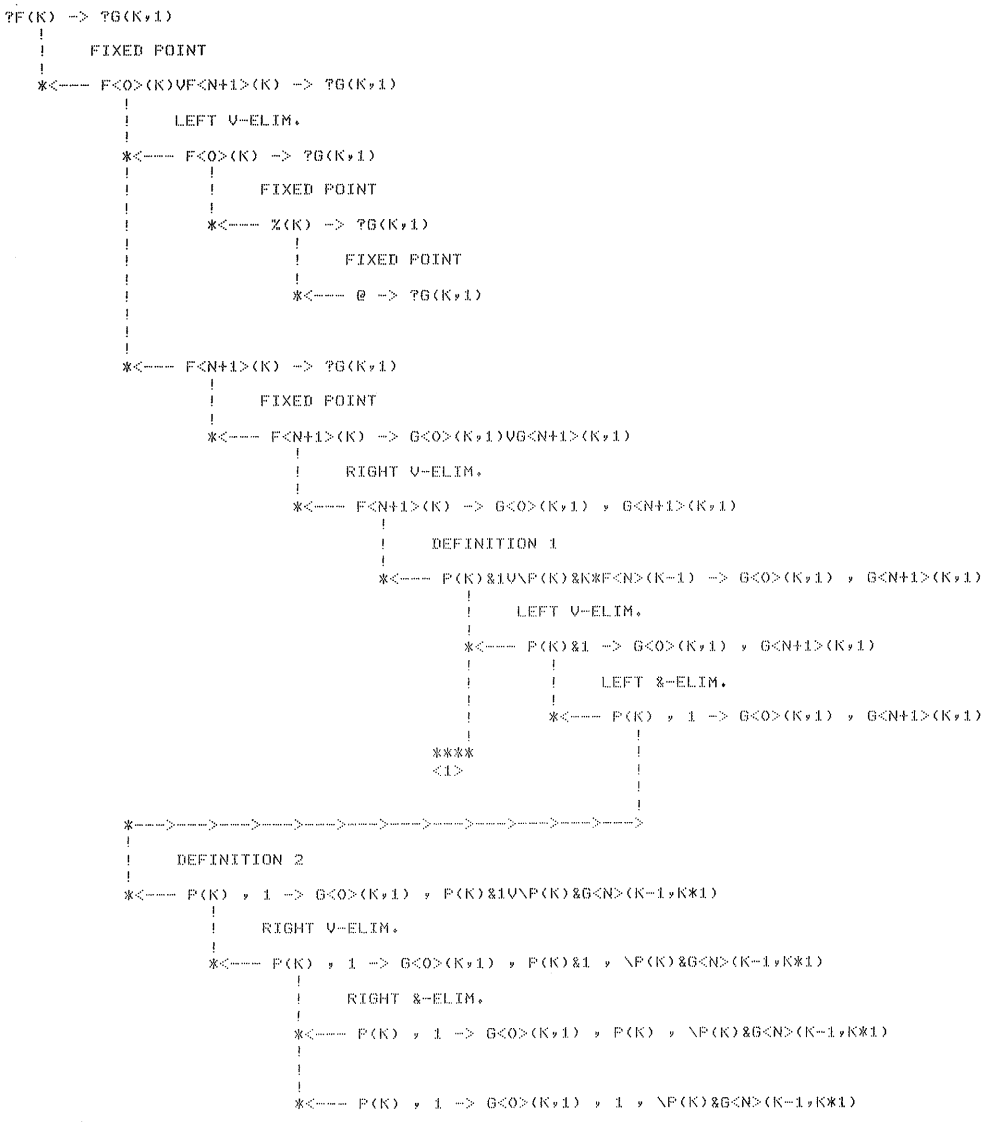
WITH INDUCTION HYPOTHESIS

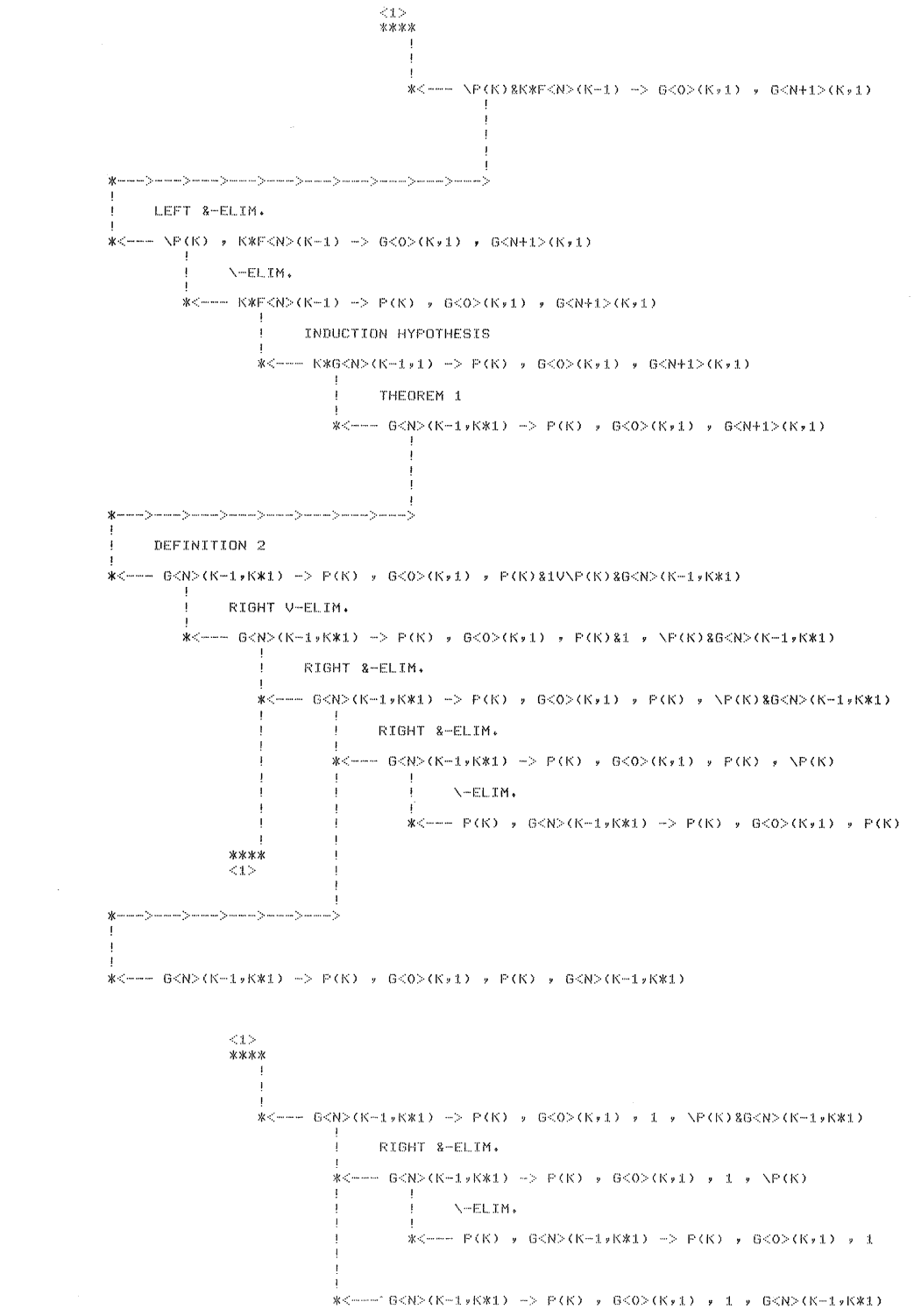
0. F<N>($K) -> G<N>($K,1)

WITH THEOREM

1. $X*G<N>($Y,$Z) -> G<N>($Y,$X*$Z)

PROOF FIGURE





THIS COMPLETES THE PROOF.