数理解析研究所講究録
第 454 巻 1982 年 61-93

MONADIC RECURSION SCHEMES WITH TWO EXITS

YUTAKA KANAYAMA

UNIVERSITY OF TSUKUBA
INSTITUTE OF INFORMATION SCIENCE
SAKURA, IBARAKI 305 JAPAN

ABSTRACT

This paper presents a new language whose describing ability is in a sense equal to monadic recursion schemes, and a formal axiom system which derives strong equivalence among monadic recursion schemes. The main feature of the K-schemes is that each scheme has one entry and two exits. Basic theorems and a few more complex examples are presented.

1. Introduction

Monadic recursion schemes have been extensively studied as models of computer programs [1][2][3][4][5]. In the class of Ianov schemes, which is a restricted class of monadic recursion schemes, the equivalence problem is solvable [6]. Friedman and others have demonstrated that the strong equivalence problem for monadic recursion schemes is decidable if and only if the equivalence problem for languages accepted by deterministic pushdown automata is decidable [3][5].

The main purposes of this papers are to propose a new method for describing monadic program schemes, and to propose a powerful axiom system, the K-system, by which the equivalence among schemes can be deduced. A similar system, μ-calculus, has already been presented by deBakker [2]. Since almost all of its axioms are proven from more elementary axioms, the K-system may be said to be a refined variation of μ-calculus.

The flavor of the K-system will be given through simple examples. Consider the following programs:

A: t:=f(t); <u>while</u> p(t) <u>do</u> t:=f(t)

B: <u>repeat</u> t:=f(t) <u>until</u> ¬p(t)

Their equivalent flowcharts are shown in Figure 1 and 2 respectively. Here t is the only program variable. The equivalence of the two algorithms could be shown by using a formal method having the power of mathematical induction. In the K-system, these algorithms are expressed as fμx(pfx) and μx(fpx) respectively and the equivalence is proven in Example 4.2.

Let us define two monadic recursion schemes F and G as follows:

$F(t) \leftarrow \underline{\text{if}}\ p(t)\ \underline{\text{then}}\ g(F(f(t)))\ \underline{\text{else}}\ g(h(t))$

$\begin{cases} G(t) \leftarrow g(G_1(t)) \\ G_1(t) \leftarrow \underline{\text{if}}\ p(t)\ \underline{\text{then}}\ g(G_1(f(t)))\ \underline{\text{else}}\ h(t) \end{cases}$

They are translated into K-language as $\mu x(pfxg+hg)$ and $(\mu x(pfxg+h))g$ respectively and the equivalence is proved also in Example 4.2.

The principal features of K-language are as follows:

(1) Every scheme can have one entry and two exits. (Thus we can say that one-exitness is not a necessary condition for structured programming.)

(2) The dual operators (·) and (+) are used instead of <;> and <<u>if</u> <u>then</u> <u>else</u>>.

(3) The negation operator (‒) is introduced.

(4) The recursive or naming operator μ is used [2].

(5) Each scheme is expressed by a single expression instead of a system of simultaneous equations.

The axiom system to be presented is, in a sense, a mixture of Boolean algebra and the system for regular expressions [7]. Although the completeness of the system is still unknown, it seems at least to be a powerful tool for the investigation of monadic recursion schemes and control flow of computer programs, because we have been successful in proving many basic equalities and some sophisticated examples.

2. Syntax

We are interested in monadic recursion schemes in a special form. That language is called K-language In this section the syntax of K-language and some syntax-related properties are defined. The property of a scheme having or not having two exits is an example of a syntactical property. Those properties about expressions are important in the deductive procedure presented in Section 3.

2.1 K-schemes

In this system, we have

(1) The set of function symbols $A = \{\mathbb{1}, \mathbb{0}, f_1, f_2, \cdots\}$.

(2) The set of predicate symbols $P = \{p_1, p_2, \cdots\}$.

(3) The set of variables $V = \{x_1, x_2, \cdots\}$.

Sometimes f, g, p, q, x or y are used instead of f_1, f_2, p_1, p_2, x_1 or x_2 respectively. The set $B = A \cup P$ is called the set of basic symbols.

Each K-scheme, or simply scheme, is constructed by basic symbols, variables, parentheses and operators $\cdot$, $+$, $-$ and μ.

Definition 3.1

Schemes are defined as follows:

(1) If $s \in B \cup V$, then s is a scheme. That is, every basic symbol or variable itself is a scheme.

(2) If F and G are schemes and x is a variable, then $(F \cdot G)$, $(F+G)$, $(-F)$ and $(\mu x F)$ are schemes.

Example 2.1 The followings are schemes.

a, $(p \cdot q)$, $(\mu x(((p \cdot f) \cdot x) + g))$, $(-((-p) + (-q)))$.

We write $F = G$ if F and G are identical strings. If G is a substring of a scheme F and F is a scheme, then we write $G \leqslant F$ and G is called a <u>subscheme</u> of F. For any scheme F, $F \leqslant F$. The number of occurrences of operators $\cdot$, +, - and μ in a scheme F is called the <u>height</u> of F and denoted by ht(F).

In a scheme (μxF), F is called a <u>scope</u> of μx. An occurrence of a variable x is said to be <u>bound</u> if it occurs immediately after μ or in a scope of μx. If an occurrence of a variable is not bound, then it is said to be <u>free</u>. A program without free occurrences of variables is said to be <u>closed</u>. A scheme G is said to be <u>free for</u> x <u>in</u> F, if no free occurrences of x in F lie within the scope of any μy, where a free y occurs in G.

<u>Example 2.2</u>

We may omit parentheses and operators by using the following rules:

(1) The operators are put in order of strength as follows: +, $\cdot$, −, μ.

(2) The dots may be omitted.

(3) The outermost parentheses may be omitted.

(4) (-F) may be written as $\bar{F}$.

Hereafter, * stands for the operator $\cdot$ or +.

<u>Example 2.3</u> Schemes shown in Example 2.1 are represented in an abbreviated form as follows:

$$a,\ pq,\ \mu x((pf)x + g),\ \overline{\bar{p} + \bar{q}}.$$

F[H/G] denotes a scheme obtained from F by replacing an occurrence of G by H.

$F[G/x]_f$ denotes a scheme obtained from F by replacing all free occurrences of x by G.

Example 2.4

If $F \equiv G \equiv pfx + g$, then $F[G/x]_f \equiv pf(pfx + g) + g$. If $H \equiv pfx + \mu x(qhx)$, then $H[G/x]_f \equiv pf(pfx + g) + \mu x(qhx)$.

2.2 Syntactical Properties

Several syntactical properties of schemes, such as the concept of entry, exit, boolean-ness and regularity, will be defined in this section.

Let G be a subscheme of F. If the relation $G \leqslant_{en} F$ is derived by using the following rules, G is said to be an entry of F[†]:

(1) $F \leqslant_{en} F$ for every F.

(2) If $G \leqslant_{en} F$, then $G \leqslant_{en} (F*H)$ and $G \leqslant_{en} \overline{F}$ for every F, G and H

Example 2.5

$F \leqslant_{en} F(GH)$, $F \leqslant_{en} (FG)H$, $F \leqslant_{en} \overline{FG}$ and $F \leqslant_{en} FG + H$.

See Propositions 4.1 and 4.2.

A skeleton of a scheme F is an approximation of F that does not contain μ operators. This concept is used in testing whether a scheme has a dot-exit or a plus-exit. A skeleton sk(F) of a scheme F is recursively defined as follows:

(1) $sk(a) = a$, if $a \in B \cup V$.

(2) $sk(F * G) \equiv sk(F) * sk(G)$.

(3) $sk(\overline{F}) \equiv \overline{sk(F)}$.

(4) $sk(\mu xF) \equiv sk(F)[sk(F)[\mathbb{0}/x]_f/x]_f$.

Example 2.6

$sk(pf + g) \equiv pf + g$

$sk(px) \equiv px$

$sk(\mu x(px)) \equiv p(p\mathbb{0})$

$sk(\mu x(p\overline{x})) \equiv p\overline{p\overline{\mathbb{0}}}$

---

† In the flowchart representation of schemes shown in Figure 3, an entry of F is a box or a set of boxes in F which is located at the upper left corner of F.

The concept of the dot exit and the plus exit of a scheme is clear if we express it in flowchart representation (See Figure 3). Strictly speaking, however, the property about the exits of schemes can be defined syntactically. If $ef^{\cdot}(F)$ or $ef^{+}(F)$ is derived by using the following rules, the scheme F is said to be <u>dot-exit free</u> or <u>plus-exit free</u> respectively:

(1) $ef^{+}(f)$, if $f \in A$.

$\quad ef^{\cdot}(\mathbb{O})$.

(2) $ef^{*}(F*G) = ef^{*}(G)$.

(3) $ef^{*}(F\bar{*}G) = ef^{*}(F) \wedge ef^{*}(G)$. ($\bar{\cdot}=+$ and $\bar{+}=\cdot$)

(4) $ef^{*}(\bar{F}) = ef^{\bar{*}}(F)$.

(5) $ef^{*}(\mu xF) = ef^{*}(sk(\mu xF))$.

Every scheme is considered to have at most two exits, the dot exit and the plus exit (see Figure 3). For example, the scheme pf has both exits. Some schemes have, however, only one exit or no exits at all. For example, pf+g does not have the plus exit.

<u>Example 2.7</u>

$ef^{+}(pf+g) = ef^{+}(g) = $ true.

$ef^{\cdot}(p\mathbb{O}) = ef^{\cdot}(\mathbb{O}) = $ true.

$ef^{\cdot}(px) = ef^{\cdot}(x) = $ false.

$ef^{\cdot}(\mu x(px)) = ef^{\cdot}(sk(\mu x(px))) = ef^{\cdot}(px)[(px)[\mathbb{O}/x]_f/x]_f)$

$= ef^{\cdot}(p(p\mathbb{O})) = ef^{\cdot}(p\mathbb{O}) = ef^{\cdot}(\mathbb{O}) = $ true.

$ef^{\cdot}(\mu x(p\bar{x})) = ef^{\cdot}(sk(\mu x(p\bar{x}))) = ef^{\cdot}((p\bar{x})[(p\bar{x})[\mathbb{O}/x]_f/x]_f)$

$= ef^{\cdot}(p\overline{p\bar{\mathbb{O}}}) = ef^{\cdot}(\overline{p\bar{\mathbb{O}}}) = ef^{+}(p\bar{\mathbb{O}}) = ef^{+}(p) \wedge ef^{+}(\bar{\mathbb{O}}) = \text{false} \wedge ef^{\cdot}(\mathbb{O}) = $ false.

We will define the "boolean" property of a scheme. If $bl^{*}(F)$ is derived by using the following rules, F is said to be <u>$*$-boolean</u>[†]:

† If $bl^{*}(F)$, then there exist no function symbols on any path from entry of F to the $*$-exit.

(1) $bl^{\cdot}(s)$, if $s \in P \cup \{\mathbb{0}, \mathbb{1}\}$.

$bl^{+}(s)$, if $s \in B$ ($= A \cup P$).

(2) $bl^{*}(F*G) = bl^{*}(F) \wedge bl^{*}(G)$.

(3) $bl^{*}(F\bar{*}G) = bl^{*}(F) \wedge bl^{\bar{*}}(F) \wedge bl^{*}(G)$.

(4) $bl^{*}(\bar{F}) = bl^{\bar{*}}(F)$.

(5) If $ef^{*}(F)$, then $bl^{*}(F)$.

<u>Example 2.8</u>

$bl^{\cdot}(p)$, $bl^{+}(p)$, $bl^{+}(pf)$, $bl^{\cdot}(\bar{p})$, $bl^{\cdot}(pq)$ and $bl^{+}(pq)$.

Suppose G is a subscheme of a scheme F. If $G \leqslant_{bl} F$ is derived by using the following rules, G is said to be <u>boolean in</u> F:

(1) $F \leqslant_{bl} F$.

(2) If $bl^{*}(F_1)$ and $G \leqslant_{bl} F_2$, then $G \leqslant_{bl} (F_1 * F_2)$.

(3) If $G \leqslant_{bl} F_1$, then $G \leqslant_{bl} (F_1 * F_2)$ and $G \leqslant_{bl} \bar{F}_1$.

<u>Example 2.9</u>

$p \leqslant_{bl} (pq)$, $p \leqslant_{bl} (qp)$ and $F \leqslant_{bl} (pq + F)$ and $H \leqslant_{bl} (pF + qG + H)$ if $ef^{+}(F)$ and $ef^{+}(G)$.

$F[H/G]_{bl}$ denotes a scheme which is obtained from F by replacing (possibly null) occurrences of G by H such that $G \leqslant_{bl} F$.

<u>Example 2.10</u>

Take $F \equiv pq$, $G \equiv pp$ and $H \equiv qp + fp$. Then $F[\mathbb{1}/p]_{bl} \equiv \mathbb{1}q$, $F[\mathbb{1}/q]_{bl} \equiv p\mathbb{1}$, $G[\mathbb{1}/p]_{bl} \equiv \mathbb{1}\mathbb{1}$, $G[\mathbb{1}/q]_{bl} \equiv pp$ and $H[\mathbb{1}/p]_{bl} \equiv q\mathbb{1} + fp$.

Consider schemes $p \cdot q$ and $pf + g$ and their flow chart equivalents. In the first, the value of the program variable t does not change during execution. On the other hand, assignment operations to t, $t := f(t)$ or $t := g(t)$, occur during the execution of the second. We will syntactically define the property of an exit of a scheme as follows:

If $rg^{\cdot}(F)$ ($rg^{\bar{+}}(F)$) is derived by using the following rules, F is said to be <u>dot-regular</u> (<u>plus-regular</u>):

(1) $rg^{\cdot}(f)$, if $f \in A - \{\mathbb{1}\}$.

$rg^{+}(f)$, if $f \in A$.

(2) $rg^{*}(F*G) = rg^{*}(F) \vee rg^{*}(G)$.

(3) $rg^{*}(F\bar{*}G) = rg^{*}(F) \wedge (rg^{\bar{*}}(F) \vee rg^{*}(G))$.

(4) $rg^{*}(\bar{F}) = rg^{\bar{*}}(F)$

(5) $rg^{*}(F)$ if $ef^{*}(F)$

<u>Example 2.11</u>

$rg^{\cdot}(pf)$, $rg^{\cdot}(fp)$, $rg^{+}(\bar{f})$. $rg^{\cdot}(pf + g)$ and $rg^{\cdot}((p + q)(pf + g))$.

See Proposition 4.3.

Assume that G is a subscheme of a scheme F. If $G \leqslant_{rg} F$ is derived by using the following rules, then G is said to be <u>regular in F</u>†:

(1) If $rg^{*}(F_1)$ and $G \leqslant F_2$, then $G \leqslant_{rg} (F_1*F_2)$.

(2) If $G \leqslant_{rg} F_1$, then $G \leqslant_{rg} (F_1*F_2)$, $G \leqslant_{rg} (F_2*F_1)$ and $G \leqslant_{rg} \bar{F}_1$.

<u>Example 2.12</u>

$F \leqslant_{rg} (fpF)$, $H \leqslant_{rg} (fpHF + G)$ and $F \leqslant_{rg} (pfFg + HG)$.

See Example 4.2.

---

† If $G \leqslant_{rg} F$, then there exists at least one function symbol on any path from the entry of F to the entry of G, or there exist no paths from the entry of F to the entry of G.

3. The Axiom System

The purpose of this section is to present the axiom system K for deducing equivalence among K-schemes. This system resembles the system of regular expressions by Salomaa [7] and the one of Boolean Algebra. Relations between program schemes and regular expressions have been discussed in many papers [8][9].

We are interestested in an equation $F = G$ between two schemes F and G. The meaning of the equation is described in Section 5. The purpose of K is to derive "valid equalities" among schemes. The next section demonstrates several equations whose two sides are quite different in form. The K system consists of eight axioms A1∼A8 and four inference rules R1∼R4. If F, G, H and J are any schemes, x is any variable, and p is any predicate symbol, then the following (A1)∼(A8) are axioms and (R1)∼(R4) are rules of inference of K. Here H[G/F] denotes a scheme which is obtained from H by replacing an occurrence of F by G.

A1 $\overline{\overline{\mathbb{1}}} = \mathbb{1}$.

A2 $\mathbb{1}F = F$.

A3 $\mathbb{1} + F = \mathbb{1}$.

A4 $\overline{\mathbb{1}} + F = F$.

A5 $\overline{\mathbb{1}}F = \overline{\mathbb{1}}$.

A6 $F = \mathbb{0}$, if $ef^{\cdot}(F)$ and $ef^{+}(F)$.

A7 $\mu xF = F[\mu xF/x]_f$, if F is free for x in F.

A8 $\mu xF = \mu x(F[\mathbb{0}/x]_{bl})$

R1(Substitution) $H[G/F] = J$ and $H[G/F] = H$ are direct consequences of $F = G$ and $H = J$.

R2(Entry) Assume that $H \leqslant_{en} F$ and $H \leqslant_{en} G$. Then $F = G$ is a direct consequence of $F[\mathbb{1}/H] = G[\mathbb{1}/H]$ and $F[\overline{\mathbb{1}}/H] = G[\overline{\mathbb{1}}/H]$.

R2^{+}(Entry) Assume that $H \leqslant_{en} F$, $H \leqslant_{en} G$ and $ef^{+}(H)$. Then $F = G$ is a direct consequence of $F[\mathbb{1}/H] = G[\mathbb{1}/H]$.

R2$^{\cdot}$(Entry) Assume that $H \leqslant_{en} F$, $H \leqslant_{en} G$ and ef$^{\cdot}$(H). Then $F = G$ is a direct consequence of $F[\bar{1}/H] = G[\bar{1}/H]$.

R3(Boolean) F = G is a direct consequence of $F[1/p]_{bl} = G[1/p]_{bl}$ and $F[\bar{1}/p]_{bl} = G[\bar{1}/p]_{bl}$.

R4(Solution of equations) Assume that F is free for x in G[x/F]rg and x is not a free variable in G. Then $F = \mu x G[x/F]_{rg}$ is a direct consequence of F = G.

An equation E is said to be a <u>consequence</u> of a set of equations **E** iff there is a sequence $E_1, \cdots, E_n$ of equations such that $E = E_n$ and, for each i, either E_i is an axiom, $E_i \in$ **E**, or E_i is a direct consequence by some rule of inference of some of the preceding schemes in the sequence. We write **E** $\vdash$ E as an abbreviation for "E is a consequence of **E**". If **E** is the empty set, we write $\vdash$E, and E is called a <u>theorem</u>.

<u>Example 3.1</u> $F_1 = F_2 \vdash F_2 = F_1$, because $F_2 = F_1$ is a direct consequence of $F_1 = F_2$ if we take $F = F_1$, $G = F_2$ and $H = F_1$ in R1.

4. Basic theorems and examples

Some important results and interesting examples derived from the K-system are presented in this section.

The substitution rule R1 and the results in the following lemma are used below without being explicitly referred to.

Lemma 4.1 If $\vdash F = G$, then $\vdash G = F$. If $\vdash F = G$ and $\vdash G = H$, then $\vdash F = H$. $\vdash F = F$. If $\vdash F = G$ and $\vdash H = J$, then $\vdash FH = GJ$, $\vdash F + H = G + J$, $\vdash \overline{F} = \overline{G}$ and $\vdash \mu x F = \mu x G$.

The proof of Lemma 1 is straightforward, by R1 and A2. The next Lemma permits renaming of bound variables as in predicate calculus.

Lemma 4.2 If F is free for x in F, F is free for y in F, all free ccurrences of x are regular in F and there exists no free occurrences of y in F, then $\vdash \mu x F = \mu y F[y/x]_{rg}$.

In the following, the dual results are shown in pairs.

Proposition 4.1

(1) $\vdash (FG)H = F(GH)$, $\vdash (F+G)+H = F+(G+H)$

(2) $\vdash \overline{FG} = \overline{F}+\overline{G}$, $\vdash \overline{F+G} = \overline{F}\,\overline{G}$ (See Figure 4)

(3) $\vdash F\mathbb{1} = F$, $\vdash F+\overline{\mathbb{1}} = F$

(4) $\vdash F+\mathbb{1} = \overline{F}+\mathbb{1}$, $\vdash F\overline{\mathbb{1}} = \overline{F}\,\overline{\mathbb{1}}$

The proof of Proposition 4.1 is by R2 and A1 ~ A5. Part (2) is similar to De Morgan's theorem in Boolean algebra.

Proposition 4.2

(1) $\vdash F+G = F$, if $ef^{+}(F)$. $\vdash FG = F$ if $ef^{\cdot}(F)$.

(2) $\vdash (F+G)H = FH+GH$ if $ef^{+}(H)$. $\vdash FG+H = (F+H)(G+H)$ if $ef^{\cdot}(H)$.

(3) $\vdash FG+H = F(G+H)+H$ if $ef^{+}(H)$. $\vdash (F+G)H = (F+GH)H$ if $ef^{\cdot}(H)$.

(See Figure 5)

(4) $\vdash (FG+H) = \overline{F}H+G$ if $ef^{+}(G)$ and $ef^{+}(H)$.

$\vdash (F+G)H = (\overline{F}+H)G$ if $ef^{\cdot}(G)$ and $ef^{\cdot}(H)$.

(5) $\vdash \overline{F}+G = FG$ if $ef^{+}(F)$.

(6) $\vdash FG+H = F(G+H)$ if $ef^{+}(F)$.

$\vdash (F+G)H = F+GH$ if $ef^{\cdot}(F)$.

The proof of Proposition 4.2 is by $R2^{+}$, $R1^{\cdot}$ and R2.

<u>Proposition 4.3</u>

(1) $\vdash pp = p$, $\vdash p+p = p$.

(2) $\vdash p\overline{p} = \overline{\mathbb{1}}$, $\vdash p+\overline{p} = \mathbb{1}$.

(3) $\vdash p+\mathbb{1} = \mathbb{1}$, $\vdash p\overline{\mathbb{1}} = \overline{\mathbb{1}}$.

(4) $\vdash pq = qp$, $\vdash p+q = q+p$.

(5) $\vdash pF+F = F$, if $ef^{+}(F)$. $\vdash (p+F)F = F$ if $ef^{\cdot}(F)$.

(6) $\vdash p(F+G) = p(pF+G)$, $\vdash p+FG = p+(p+F)G$. (See Figure 6)

(7) $\vdash pF+pG = pF$ if $ef^{+}(F)$, $\vdash (p+F)(p+G) = p+F$ if $ef^{\cdot}(F)$.

(8) $\vdash pqF+pG+qH = pqF+qH+pG$, if $ef^{+}(F)$, $ef^{+}(G)$ and $ef^{+}(H)$.

$\vdash (p+q+F)(p+G)(q+H) = (p+q+F)(q+H)(p+G)$, if $ef^{\cdot}(F)$, $ef^{\cdot}(G)$ and $ef^{\cdot}(H)$.

(9) $\vdash p(qF_1+F_2)+qF_3+F_4 = q(pF_1+F_3)+pF_2+F_4$ if $ef^{+}(F_1)$, $ef^{+}(F_2)$, $ef^{+}(F_3)$ and $ef^{+}(F_4)$.

$\vdash (p+(q+F_1)F_2)(q+F_3)F_4 = (q+(p+F_1)F_3)(p+F_2)F_4$ if $ef^{\cdot}(F_1)$, $ef^{\cdot}(F_2)$, $ef^{\cdot}(F_3)$ and $ef^{\cdot}(F_4)$.

Proof. (1) The proof of the first equation is:

(a) $\mathbb{1}\mathbb{1} = \mathbb{1}$ A2.

(b) $\overline{\mathbb{1}}\overline{\mathbb{1}} = \overline{\mathbb{1}}$ A5.

(c) $pp = p$ (a), (b) and R3.

The remaining proofs are omitted.

<u>Proposition 4.4</u> Assume that $ef^{+}(F)$ and $ef^{+}(G)$. Then $pq = \overline{\mathbb{I}}$ $\vdash pF+qG = qG+pF$.

Proof. Assume that H is an arbitrary scheme such that $ef^{+}(H)$.

(1)	$pqH+pF+qG = pqH+qG+pF$	Proposition 4.3(8).
(2)	$pq = \overline{\mathbb{I}}$	Hypothesis.
(3)	$\overline{\mathbb{I}}H+pF+qG = \overline{\mathbb{I}}H+qG+pF$	(1), (2), Lemma 4.1.
(4)	$pF+qG = qG+pF$	(3), A5, A4.

Q.E.D.

Hereafter, the set of theorems $\vdash F_1 = F_2$, $F_2 = F_3$, $\cdots$, $\vdash F_{n-1} = F_n$ are denoted by $F_1 = F_2 = \cdots = F_n$.

<u>Example 4.1</u>

(1) If $F \equiv \mu x(pfx+g)$, then

$$\begin{aligned}\vdash F &= pfF+g\\ &= pf(pfF+g)+g\\ &= pf(pf(pfF+g)+g)+g\\ &= \cdots\end{aligned}$$

(2) If $G \equiv \mu x(pfxg+h)$, then

$$\begin{aligned}\vdash G &= pfGf+h\\ &= pf(pfGg+h)g+h\\ &= pf(pf(pfGg+h)g+h)g+h\\ &= \cdots\end{aligned}$$

(3) $$\begin{aligned}\vdash H &\equiv \mu x(pfxx+g)\\ &= pfHH+g\\ &= pf(pfHH+g)(pfHH+g)+g\\ &= \cdots\end{aligned}$$

(4) $$\begin{aligned}\vdash J &\equiv \mu x(pf\overline{x}+g)\\ &= pf\overline{J}+g\\ &= pf(\overline{pf\overline{J}+g})+g\\ &= pf(\overline{p}+\overline{f}+J)\overline{g}+g\\ &= \cdots\end{aligned}$$

Example 4.2 Several pairs of equivalent schemes whose recursive structures are not the same are shown here. They are derived by using R4.

(1) $\vdash f\mu x(pfx) = \mu x(fpx)$. This is because $\vdash F \equiv f\mu x(pfx) = f(pf\mu x(pfx))$
$= fp(f\mu x(pfx)) \equiv fpF$; Therefore $\vdash F = \mu x(fpx)$ by R4.

(2) $\vdash f\mu x(pfxF+G) = \mu x(fpxF+G)$, where F, G are arbitrary schemes.
This is because, $\vdash H \equiv f\mu x(pfxF+G) = f(pf\mu x(pfxF+G)F+G)$
$= fpf\mu x(pfxF+G)F+G \equiv fpHF+G$; therefore $\vdash H = \mu x(fpxF+G)$.

(3) $\vdash \mu x(pf\bar{x})g+h = \mu x(pf(pfx+g)+h)$.
Take $F \equiv \mu x(pf\bar{x})g+h$. Then $\vdash F = pf\overline{\mu x(pf\bar{x})}g+h = pf\overline{pf\overline{\mu x(pf\bar{x})}}g+h$
$= pf(\bar{p}+\bar{f}+\mu x(pf\bar{x}))g+h = pf(\bar{p}+f\mu x(pf\bar{x}))g+h = pf(\bar{p}g+f\mu x(pf\bar{x})g)+h$
$= pf(\bar{p}g+f\mu x(pf\bar{x})g+h)+h = pf(p(f\mu x(pf\bar{x})g+h)+g)+h$
$= pf(pf(\mu x(pf\bar{x})g+h)+g)+h \equiv pf(pfF+g)+h$.
Therefore, by R4, $\vdash F = \mu x(pf(pfx+g)+h)$.

(4) $\vdash \mu x(pfxg+hg) = (\mu x(pfxg+h))g$. The idea of this equivalence is based on the example by Korenjack and Hopcroft [10]. (See Figure 7) Take $F \equiv (\mu x(pfxg+h))g$. Then $\vdash F = (pf\mu x(pfxg+h)g+h)g$
$= pf\mu x(pfxg+h)gg+hg \equiv pfFg+hg$. Hence $\vdash F = \mu x(pfxg+hg)$ by R4.

We will show a final example taken from Dijkstra [11].

Example 4.3

Consider the following two programs P_1 and P_2, both of which evaluate the greatest common divisor of two natural numbers.

P_1: <u>while</u> $a \neq b$ <u>do</u> <u>if</u> $a > b$ <u>then</u> $a := a-b$ <u>else</u> $b := b-a$

P_2: <u>while</u> $a \neq b$ <u>do</u> <u>begin</u> <u>while</u> $a > b$ <u>do</u> $a := a-b$
<u>while</u> $b > a$ <u>do</u> $b := b-a$ <u>end</u>

They are rewritten as

P'_α : <u>while</u> $p \vee q$ <u>do</u> <u>if</u> p <u>then</u> f <u>else</u> g

P'_β : <u>while</u> $p \vee q$ <u>do</u> <u>begin</u> <u>while</u> p <u>do</u> f; <u>while</u> q <u>do</u> g <u>end</u>,
where $p \wedge q =$ <u>false</u>.

These programs are translated into K-schemes as follows:

$F \equiv \mu x((p+q)(pf+g)x+\mathbb{1})$

$G \equiv \mu x((p+q)\mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})x+\mathbb{1})$

We want to show that $pq=\bar{\mathbb{1}} \vdash F=G$. Let $H \equiv \mu x(pfx+qgx+\mathbb{1})$. (See Figure 8) First we demonstrate that $pq=\bar{\mathbb{1}} \vdash F=H$, and later that $pq=\bar{\mathbb{1}} \vdash G=H$.

(a) Proof that $pq=\bar{\mathbb{1}} \vdash F=H$.

$\vdash F \equiv \mu x((p+q)(pf+g)x+\mathbb{1}) = (p+q)(pf+g)F+\mathbb{1} = (p(pf+g)+q(pf+g))F+\mathbb{1}$

$= (p(f+g)+q(qpf+g))F+\mathbb{1} = (pf+q(\bar{\mathbb{1}}f+g))F+\mathbb{1} = (pf+qg)F+\mathbb{1}$

$= pfF+qgF+\mathbb{1}.$

Therefore, $\vdash F = \mu x(pfx+qgx+1) \equiv H.$

(b) Proof that $pq=\bar{\mathbb{1}} \vdash G=H$.

$\vdash G \equiv \mu x((p+q)\mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})x+\mathbb{1}) = (p+q)\mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= (p\mu x(pfx+\mathbb{1})+q\mu x(pfx+\mathbb{1}))\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= (p(pf\mu x(pfx+\mathbb{1})+\mathbb{1})+q(pf\mu x(pfx+\mathbb{1})+\mathbb{1}))\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= (p(f\mu x(pfx+\mathbb{1})+\mathbb{1})+q(qpf\mu x(pfx+\mathbb{1})+\mathbb{1})\mu x(qgx+1)G+1$

$= (pf\mu x(pfx+\mathbb{1})+q(\bar{\mathbb{1}}f\mu x(pfx+\mathbb{1})+\mathbb{1})\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= (pf\mu x(pfx+\mathbb{1})+q)\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= pf\mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})G+q\mu x(qgx+\mathbb{1})G+\mathbb{1} = pfG_1+q(qg\mu x(qgx+\mathbb{1})+\mathbb{1})G+\mathbb{1}$

$= pfG_1+q(g\mu x(qgx+\mathbb{1})+\mathbb{1})G+\mathbb{1} = pfG_1+qg\mu x(qgx+\mathbb{1})G+\mathbb{1}$

$= pfG_1+qgG_2+\mathbb{1},$

where $G_1 \equiv \mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})G$ and $G_2 \equiv \mu x(qgx+\mathbb{1})G$. Therefore,

$\vdash G_1 \equiv \mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})G = (pf\mu x(pfx+\mathbb{1})+\mathbb{1})\mu x(qgx+\mathbb{1})G$

$= pf\mu x(pfx+\mathbb{1})\mu x(qgx+\mathbb{1})G+\mu x(qgx+\mathbb{1})G = pfG_1+(qg\mu x(qgx+\mathbb{1})+\mathbb{1})G$

$= pfG_1+qg\mu x(qgx+\mathbb{1})G+G = pfG_1+qgG_2+pfG_1+qgG_2+\mathbb{1}$

$= pfG_1+pfG_1+qgG_2+qgG_2+\mathbb{1} = pfG_1+qgG_2+\mathbb{1} = G$

$\vdash G_2 \equiv \mu x(qgx+\mathbb{1})G = (qg\mu x(qgx+\mathbb{1})+\mathbb{1})G = qg\mu x(qgx+\mathbb{1})G+G$

$= qgG_2+pfG_1+qgG_2+\mathbb{1} = pfG_1+qgG_2+qgG_2+\mathbb{1} = pfG_1+qgG_2+\mathbb{1} = G$

Hence $\vdash G_1 = G_2 = G$, and $\vdash G = pfG+qgG+\mathbb{1}$. Therefore $\vdash G = \mu x(pfx+qgx+\mathbb{1}) \equiv H.$

5. Semantics

The meaning of a scheme in the K–language is defined in such a way that the class of all interpreted funcitons from the K-schemes includes the class of all interpreted functions from monadic recursion schemes [4][5].

5.1. Definitions

Let D be any non-empty set. We add a special element $\perp$ to D to obtain the set $D_\perp = D \cup \{\perp\}$. A partial order $\sqsubseteq$ is defined on $D_\perp$ such that $s \sqsubseteq t$ iff $s = \perp$ or $s = t$. If ϕ is a total function: $D \to D$, then it is extended to the function: $D_\perp \to D_\perp$ such that $\phi(\perp) = \perp$ [12][13].

The least upper bound operation $\sqcup$ is defined as follows: $t \sqcup \perp = \perp \sqcup t = t \sqcup t = t$ for all $t \in D_\perp$. $s \sqcup t$ is undefined if $s \neq \perp$, $t \neq \perp$ and $s \neq t$. $\bigsqcup_{n=0}^{\infty} t_n = t$ iff $t_n = t$ or $t_n = \perp$ for all $n \geqslant 0$, and there exists n such that $t_n = t$. The least upper bound $\phi \sqcup \psi$ of functions ϕ and ψ: $D_\perp \to D_\perp$ is defined by $(\phi \sqcup \psi)(t) = \phi(t) \sqcup \psi(t)$ for any $t \in D_\perp$. This operation can be extended for the class of enumerable functions $\phi_0, \phi_1, \ldots$.

Let D^D and 2^D denote the set of all total functions: $D \to D$ and the set of all total predicates: $D \to \{\text{true, false}\}$ respectively. An interpretation I is a quadruple $(D, \mathbb{A}, \mathbb{P}, \mathbb{V}) = (D, \mathbb{A}, \mathbb{P}, (\mathbb{V}^\cdot, \mathbb{V}^+))$. where $\mathbb{A}$ is a mapping: $A \to D^D$, $\mathbb{P}$ a mapping: $P \to 2^D$, and $\mathbb{V}$ a mapping: $V \to D_\perp \times D_\perp$ with a condition that $\mathbb{V}^\cdot(x) = \perp$ or $\mathbb{V}^+(x) = \perp$ for any x.

Assume $I = (X, \mathbb{A}, \mathbb{P}, \mathbb{V})$. Let us define that $I[(t^\cdot, t^+)/x]$ stands for an interpretation $I' = (D', \mathbb{A}', \mathbb{P}', \mathbb{V}')$ such that $D' = D$, $\mathbb{A}' = \mathbb{A}$, $\mathbb{P}' = \mathbb{P}$ and

$$\mathbb{V}'(y) = \begin{cases} \mathbb{V}(y), & \text{if } y \neq x \\ (t^\cdot, t^+), & \text{if } y = x \end{cases}$$

$\mathbb{V}$ defines the meaning of free variables. In general, however, the role of $\mathbb{V}$ is less important than that of $\mathbb{A}$ and $\mathbb{P}$.

The meaning F_I of a scheme F under an interpretation I is a function: $D_\perp \to D_\perp \times D_\perp$. That is, for any $t \in D_\perp$, $F_I(t)$ is a pair $(F_I^{\cdot}(t), F_I^{+}(t))$. The function $F_I^{\cdot}$ stands for the dot effects of the all paths of the scheme F between the entry and the exit; the function F_I^{+} is defined in the same way with respect the plus exit.

We stipulate that E and $\perp$ denote the identity functions and the bottom function on D; i.e., $E(t) = t$ and $\boldsymbol{\perp}(t) = \perp$ for all $t \in D_\perp$. The composition $\phi \circ \psi$ of functions ϕ and ψ: $D_\perp \to D_\perp$ is given by a definition; $(\phi \circ \psi)(t) = \psi(\phi(t))$.

Each scheme is recursively interpreted as follows:

(5.1) $\mathbb{0}_I(t) = (\perp, \perp) = (\boldsymbol{\perp}, \boldsymbol{\perp})(t)$

(5.2) $\mathbb{1}_I(t) = (t, \perp) = (E, \boldsymbol{\perp})(t)$

(5.3) $f_i(t) = (\mathbb{A}(f)(t), \perp) = (\mathbb{A}(f), \boldsymbol{\perp})(t)$

(5.4) $$p_I(t) = \begin{cases} (t, \perp), & \text{if } \mathbb{P}(p)(t) \\ (\perp, t), & \text{if } \neg\mathbb{P}(p)(t) \end{cases}$$

(5.5) $x_I(t) = (\mathbb{V}^{\cdot}(x), \mathbb{V}^{+}(x)) = \mathbb{V}(x)$

(5.6) $$(FG)_I(t) = (G_I^{\cdot}(F_I^{\cdot}(t)), F_I^{+}(t) \sqcup G_I^{+}(F_I^{\cdot}(t))) = (F_I^{\cdot} \circ G_I^{\cdot}, F_I^{+} \sqcup (F_I^{\cdot} \circ G_I^{+}))(t)$$

(5.7) $$(F+G)_I(t) = (F_I^{\cdot}(t) \sqcup G_I^{\cdot}(F_I^{+}(t)). G_I^{+}(F_I^{+}(t))) = (F_I^{\cdot} \circ (F_I^{+} G_I^{\cdot}), F_I^{+} \circ G_I^{+})(t)$$

(5.8) $\bar{F}_I(t) = (F_I^{+}(t), F_I^{\cdot}(t)) = (F_I^{+}, F_I^{\cdot})(t)$

(5.9) $(\mu x F)_I(t) = \bigsqcup_{n=0} F_{I,x,n}(t)$,

$$\text{where } \begin{cases} F_{I,x,0}(t) = (\perp, \perp) = \mathbb{0}_I(t) \\ F_{I,x,n+1}(t) = F_{I[F_{I,x,n}(t)/x],x,n}(t); \; n=0,1,2,\ldots \end{cases}$$

We may interpret each K-scheme as a flowchart which has at most two exits. That is shown in Figure 3.

Lemma 5.1 If G is free for x in F, then $F_{I[G_I(t)/x]}(t) = (F[G/x]_f)_I(t)$ for all F, G, t, x and I.

We write $F \cong G$ if $F_I = G_I$ for all interpretation I.† We want to demonstrate that the axiom system K is consistent. It is helpful if we can fix on one special domain in proving validity of the system. A special class of interpretation, Herbrand interpretations, is introduced here.

The domain D is called the Herbrand universe H_K of K if D consists of the strings,

$$\lambda,\ f_1,\ f_2,\ \ldots,\ f_1f_1,\ f_1f_2,\ \ldots,\ f_2f_1,\ \ldots,$$

and $\perp$, where λ denotes the empty string. Assume that $\perp t = t\perp = \perp$ for all $t \in H_K$. An interpretation I is called Herbrand if $D = H_K$ and $f_I(t) = tf$ for any $t \in H_K$ and $f \in A$.†† Hereafter we will treat only Herbrand interpretations, because

Proposition 5.2[14] $F \cong G$ iff $F_I = G_I$ for all Herbrand interpretations I.

Example 5.1 Let us show how pf+g, q(pf+g) and $\mu x(pfx)$ are interpreted under an Herbrand interpretation I.

$$(pf+g)_I(t) = \begin{cases} (tf, \perp), & \text{if } \mathbb{P}(p)(t) \\ (tg, \perp), & \text{if } \neg\mathbb{P}(p)(t) \end{cases}$$

$$(\mu x(pfx))_I(t) = \begin{cases} (\perp,\ tf^n), & \text{if } \bigwedge_{m=0}^{n-1}\mathbb{P}(p)(tf^m) \wedge (\neg\mathbb{P}(p)(tf^n)) \\ (\perp, \perp), & \text{if } \bigwedge_{m=0}^{\infty}\mathbb{P}(p)(tf^m) \end{cases}$$

$$(q(pf+g))_I(t) = \begin{cases} (tf, \perp), & \text{if } \mathbb{P}(q)(t) \wedge \mathbb{P}(p)(t) \\ (tg, \perp), & \text{if } \mathbb{P}(q)(t) \wedge \neg\mathbb{P}(p)(t) \\ (\perp,\ t), & \text{if } \neg\mathbb{P}(q)(t) \end{cases}$$

---

† $F_I = G_I$ means that $F_I(t) = F_I(t)$ for all $t \in D_\perp$.

†† $\mathbb{P}$ and $\mathbb{V}$ are, however, not fixed.

5.2 Validity of Axioms

First, we show the validity of the elementary axioms.

Proposition 5.3 For any scheme F,

(1) $\bar{\bar{\mathbb{I}}} \cong \mathbb{1}$

(2) $\mathbb{1}F \cong F$ (3) $\mathbb{1} + F = \mathbb{1}$

(4) $\bar{\mathbb{I}} + F = F$ (5) $\bar{\mathbb{I}}F = \bar{\mathbb{I}}$

Second, we show the validity of the axiom about exits.

Lemma 5.4 $sk(F[G/x]_f) \equiv sk(F)[sk(G)/x]_f$ for any F, G and x, if G is free for x in F.

Let us define $F\langle x, n\rangle$ for n=0, 1, 2, ... as follows:

(5.10) $F\langle x, 0\rangle \equiv 0$

(5.11) $F\langle x, n+1\rangle \equiv F[F\langle x, n\rangle/x]_f$, for n=0, 1, 2, ...

Lemma 5.5 If F is free for x in F, then $F\langle x, n\rangle_I \equiv F_{I,x,n}$ for all F, x, n and I.

Lemma 5.6 If F is free for x in F, then $sk(F)\langle x, n\rangle = sk(F\langle x, n\rangle)$ for all F, x and n.

Lemma 5.7 If F has no μ-operators and $ef^*(G) \rightarrow ef^*(H)$, then $ef^*(F[G/x]_f) \rightarrow ef^*(F[H/x]_f)$ for all F, G, H, x and *. Hence, if F has no μ-operators and $ef^*(G) \leftrightarrow ef^*(H)$, then $ef^*(F[G/x]_f) \leftrightarrow ef^*(F[H/x]_f)$.

Lemma 5.8 If F has no μ-operators, then $ef^*(F\langle x, n+1\rangle) \rightarrow ef^*(F\langle x, n\rangle)$ for all F, x and n.

Lemma 5.9 If F has no μ-operators, then $ef^*(F\langle x, n\rangle) \leftrightarrow ef^*(F\langle x, 2\rangle)$ for all $n \geqslant 3$.

Lemma 5.10 If F is free for any free x in F and $ef^*(F)$, then $F_I{}^*(t) = \perp$ for all F, t, * and I.

Proposition 5.11 If $ef^{\cdot}(F)$ and $ef^{+}(F)$, then $F \cong \mathbb{0}$.

Thus, the axioms A1 to A6 are valid.

In order to demonstrate the validity of A7, we need to define the monotonic and continuous properties of schemes [12][13].

A function ϕ: $D_\perp \to D_\perp$ is said to be monotonic, when, if $s \sqsubseteq t$, then $\phi(s) \sqsubseteq \phi(t)$ for all s and t. F_I is also said to be monotonic if $F_I^{\cdot}$ and F_I^{+} are monotonic.

Lemma 5.12 For any F and I, F_I is monotonic.

Lemma 5.13 $F_{I,x,n} \sqsubseteq F_{I,x,n+1}$ for all I, x, F and n.

A function ϕ: $D_\perp \to D_\perp$ is said to be continuous if $s_0 \sqsubseteq s_1 \sqsubseteq \dots \sqsubseteq s_m \dots$, then $\bigsqcup_{m=0}^{\infty} \phi(s_m) = \phi(\bigsqcup_{m=0}^{\infty} s_m)$ for all $s_0, s_1, \dots$.

Proposition 5.14 If F is free for x in F, then $\mu xF \cong F[\mu xF/x]_f$ for any F and x.

Lemma 5.15 If bl*(G), then $G*F[H/x]_{bl} \cong G*F[(G*H)/x]_{bl}$ for any F, G, H, * and x.

Lemma 5.16 $F[F[\mathbb{0}/x]_{bl}[G/x]_f/x]_{bl} \cong F[0/x]_{bl}$ for any F, G, * and x.

Proposition 5.17 $\mu xF \cong \mu x(F[\mathbb{0}/x]_{bl})$ for any F and x.

5.3 Validity of Rules

We will show that the rules of inference in K-system preserve validity of equations.

Proposition 5.18 If $F \cong G$, then $H[G/F] \cong H$ for any F, G and H.

Lemma 5.19 If $G \leqslant_{en} F$, then $F_I^* = G_I^{\cdot} \circ F[\mathbb{1}/G]_I^* \sqcup G_I^{+} \circ F[\mathbb{1}/G]_I^*$ for any F, G, * and I, where $\circ$ means the concatenation operation of strings.

Proposition 5.20 If $H \leqslant_{en} F$, $H \leqslant_{en} G$, $F[\mathbb{1}/H] \cong G[\mathbb{1}/H]$, and $F[\bar{\mathbb{1}}/H] = G[\bar{\mathbb{1}}/H]$, then $F \cong G$.

Proposition 5.20$^+$ If $H \leqslant_{en} F$, $H \leqslant_{en} G$, $ef^+(H)$ and $F[\mathbb{1}/H] \cong G[\mathbb{1}/H]$, then $F \cong G$.

Proposition 5.20' If $H \leqslant_{en} F$, $H \leqslant_{en} G$, $ef^{\cdot}(H)$ and $F[\bar{\mathbb{I}}/H] \cong G[\bar{\mathbb{I}}/H]$, then $F \cong G$.

Lemma 5.21

$$F_I{*}(t) = \begin{cases} F[\mathbb{I}/p]_{bl}{*}(t), & \text{if } \mathbb{P}(p)(t) \\ F[\bar{\mathbb{I}}/p]_{bl}{*}(t), & \text{if} \sim \mathbb{P}(p)(t) \end{cases}$$

for any F, I, $*$, t and $p \in P$.

Proposition 5.22 If $F[\mathbb{I}/p]_{bl} \cong G[\mathbb{I}/p]_{bl}$ and $F[\bar{\mathbb{I}}/p]_{bl} \cong G[\bar{\mathbb{I}}/p]_{bl}$, then $F \cong G$.

We have a one-side result for R4 as follows:

Lemma 5.23 If F is free for x in G[x/F], x is not a free variable of G and $F \cong G$, then $\mu x G[x/F]_I \sqsubseteq F_I$ for any F, G, x and I.

We have to prove the converse using the notion of length.

The generalized Herbrand universe of K is the set of all strings GH_k generated by $A \cup \{P^{\cdot}, P^{+} | p \in P\}$ and $\perp$. Suppose we are given a generalized Herbrand interpretation $I = (GH_K, \mathbb{A}, \mathbb{P}, \mathbb{V})$. In case $t \neq \perp$, let lg(t) denote the number of occurrences of function symbols in t.

In a generalized Herbrand interpretation, we redefine the semantics (5.3) and (5.4) as follows:

(5.3)' $f_I(t) = (tf, \perp)$

(5.4)' $p_I(t) = (tp^{\cdot}, tp^{+})$

We assume that GH_K is closed under the l,u,b, operation $\sqcup$ among strings.

Example 5.2 Schemes pf+g, q(pf+g) and $\mu x(pfx)$ in Example 5.1 are interpreted under a generalized Herbrand interpretation as follows:

$$(fp+g)_I(t) = (tp^{\cdot}f \sqcup tp{+}g, \perp)$$

$$(q(pf+g))_I(t) = (tq^{\cdot}p^{\cdot}f \sqcup tq^{\cdot}p^{+}g, tg^{+})$$

$$(\mu x(pfx))_I(t) = (\perp, \bigsqcup_{n=0}^{\infty} t(p^{\cdot}f)^n p^{+})$$

In a generalized Herbrand interpretation I, if we define

$$(5.10)\quad tp^{\cdot} = \begin{cases} t, & \text{if } \mathbb{P}(p)(t) \\ \bot, & \text{if } \neg\mathbb{P}(p)(t) \end{cases}$$

$$(5.11)\quad tp^{+} = \begin{cases} t, & \text{if } \neg\mathbb{P}(p)(t) \\ \bot, & \text{if } \mathbb{P}(p)(t), \end{cases}$$

then the semantics of the K-schemes are the same as those defined in Section 5.1.

If t is a string in GH_K, let lg(t) denote the number of occurrences of function symbols in t. Suppose $u \in GH_K$ is the l,u,b, of a set S_u of strings in GH_K. Then $u_{(k)}$ denotes $\bigsqcup\{t \in S_u \mid lg(t)=k\}$, Clearly $u = \bigsqcup_{k=0}^{\infty} u_{(k)}$.

<u>Lemma 5.24</u> $\bigsqcup_{m\leq k}(st)_{(m)} \sqsubseteq \bigsqcup_{m\leq k} s_{(m)} \bigsqcup_{m\leq k} t_{(m)}$ for any s, t, and k.

<u>Lemma 5.25</u> $\bigsqcup_{m\leq k} F_I(t)_{(m)} \sqsubseteq F_I(\bigsqcup_{m\leq k} t_{(m)})$ for any F, t, k and I.

<u>Lemma 5.26</u> If rg*(F), then

$$\begin{cases} F_I{}^*(t)_{(0)} = \bot \\ \bigsqcup_{m\leq k+1} F_I{}^*(t)_{(m)} \sqsubseteq F_I{}^*(\bigsqcup_{m\leq k} t_{(m)}) \qquad k=0,\ 1,\ 2,\ \ldots \end{cases}$$

for any F, t, * and I.

<u>Lemma 5.27</u> If F is free for x in G[x/F], then $\bigsqcup_{m\leq k} G_I(t)_{(m)} \sqsubseteq (G[x/F]_{rg})_{I[\bigsqcup_{m\leq k-1} F_I(t)_{(m)}/x]}(t)$, k=1, 2, ... for any F, G, x, I and t.

<u>Proposition 5.28</u> If F is free for x in G[x/F], x is not a free variable in F and $F \cong G$, then $F \cong \mu x G[x/F]_{rg}$ for any F, G, x.

This concludes the proof that all rules on inferences R1~R4 preserves the validity of equations.

Acknowledgement

The author thanks Professor J. W. Higgins of Tsukuba University for his useful suggestions and critical reading of the manuscript.

References

[1] deBakker, J. W., and D. Scott, "A Theory of Programs", unpublished memo, Vienna, August 1969.

[2] deBakker, J. W., "Recursive Procedures", Math. Center Tracts No. 24, Amsterdam, 1971.

[3] Garland, S. J., and D. C. Luckham, "Program Schemes, Recursion Schemes, and Formal Languages", J. Computer and System Sciences, Vol. 7, pp. 119-160, 1973.

[4] Ashcroft, E., Z. Manna, and A. Pnueli, "Decidable Properties of Monadic Functional Schemes", J. ACM, Vol. 20, pp. 489-499, 1973.

[5] Friedman, E. P., "Equivalence Problems for Deterministic Context-Free Languages and Monadic Recursion Schemes", J. of Computer and System Sciences, Vol. 14, pp. 344-359, 1977.

[6] Ianov, Y. I., "The Logical Schemes of Algorithms", in Problems of Cybernetics, Vol. 1, pp. 82-140, Pergamon Press, New York, 1960.

[7] Salomaa, A., "Two Complete Axiom Systems for the Algebra of Regular Events", J. of ACM, VOl. 13, pp. 158-169, 1966.

[8] Ito, T., "Some Formal Properties of a Class of Program Schemata", Proc. IEEE Symposium on Switching and Automata Theory, 1968.

[9] Kaplan, D. M., "Regular Expressions and the Equivalence of Programs", J. of Computer and System Sciences, Vol. 3, pp. 361-386, 1969.

[10] Korenjak, A. J., and J. E. Hopcroft, "Simple Deterministic Languages", IEEE Conf. Record of 7th Annual Symp. on Switching and Automata Theory, pp. 36-46, 1966.

[11] Dijkstra, E. W., "A Short Introduction to the Art of Programming".

[12] Scott, D. S., "The Lattice of Flow Diagrams", Symposium on Semantics of Algorithmic Languages, in Lecture Notes in Mathematics-No. 188, pp. 311-366, Springer-Verlag, 1971.

[13] Manna, Z., S. Ness, and J. Vuillmin, "Inductive Methods for Proving Properties of Programs", Proc. of Conference on Proving Assertions about Programs, pp. 27-50, New Mexico State University, 1972.

[14] Luckham, D. C., D. M. R. Park, and M. S. Paterson, "On Formalized Computer Programs", J. of Computer and Systems Science, Vol. 4, pp. 220-249, 1970.

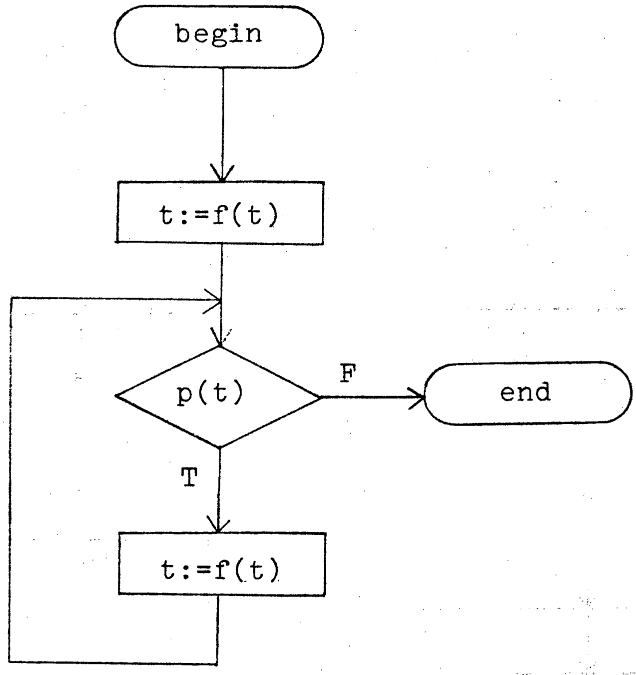

Figure 1   Program A

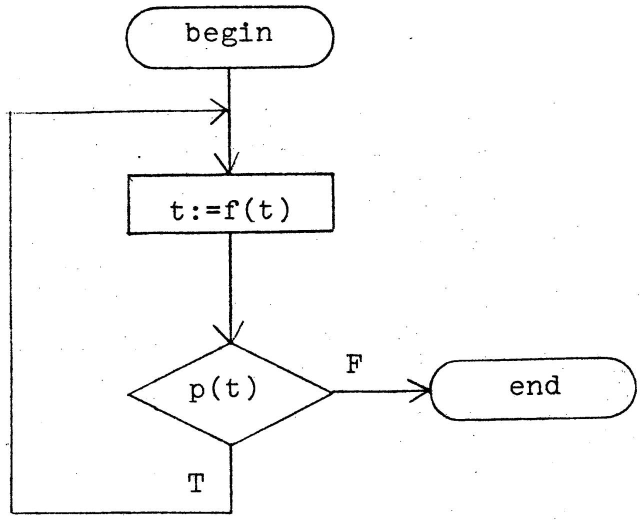

Figure 2   Program B

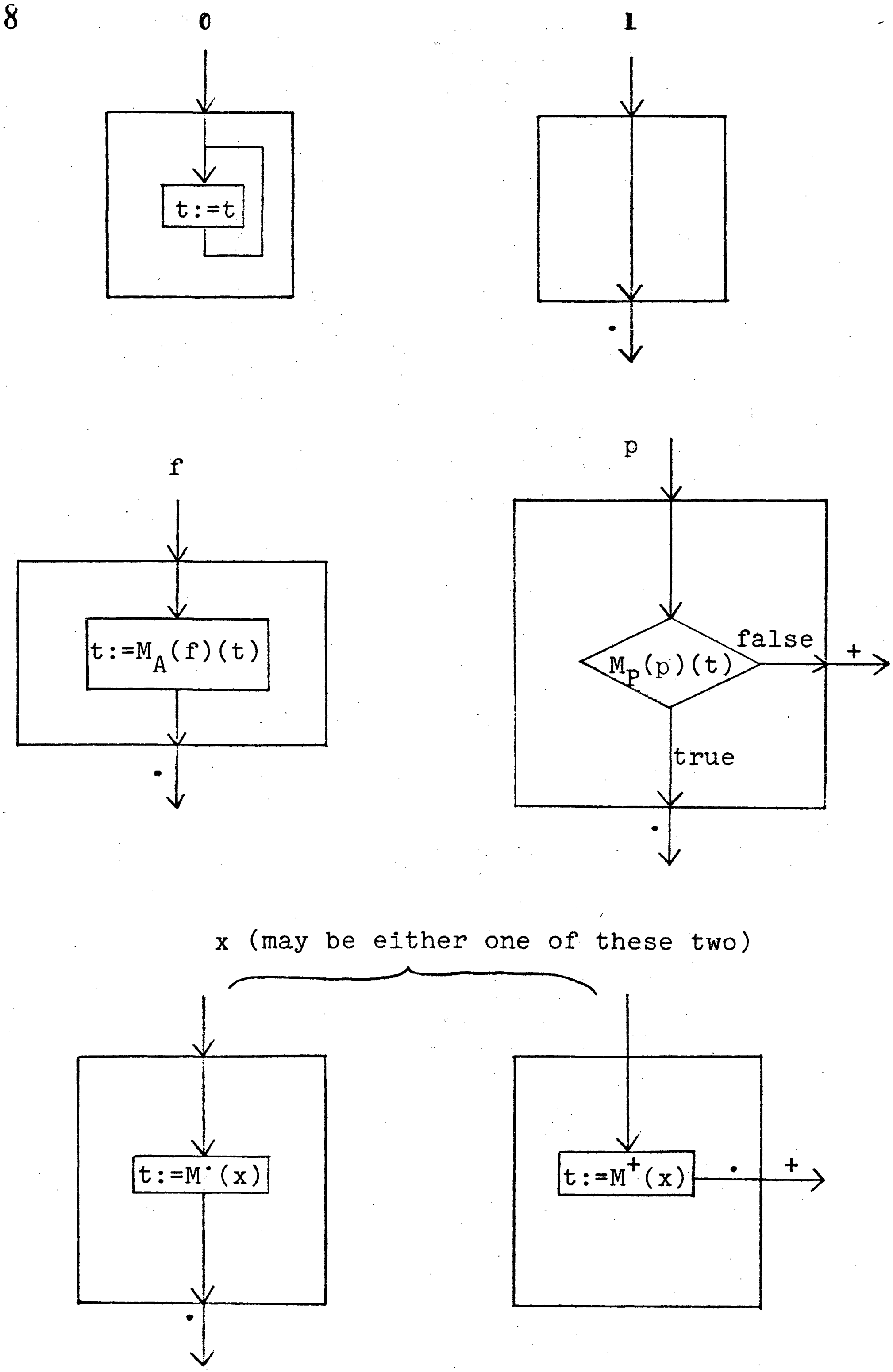

Figure 3 (a) An Instinctive Interpretation of Schemes

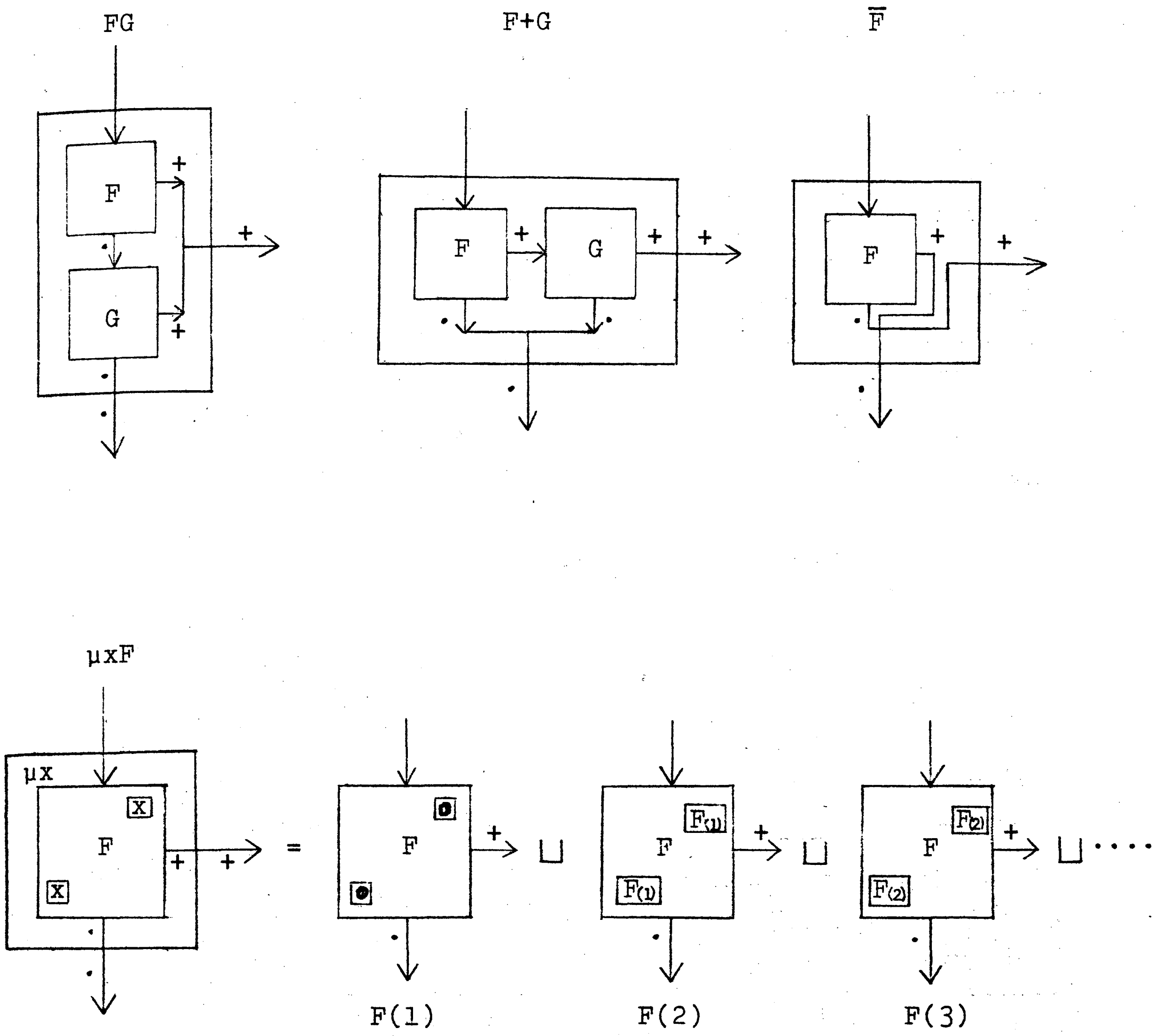

Figure 3 (b) An Instinctive Interpretation of Schemes

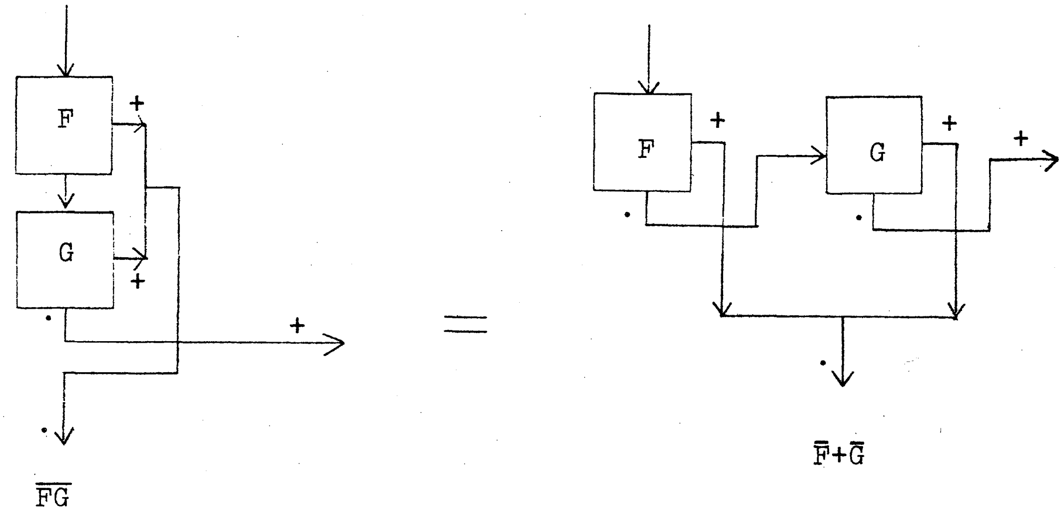

Figure 4 $\overline{FG}=\overline{F}+\overline{G}$

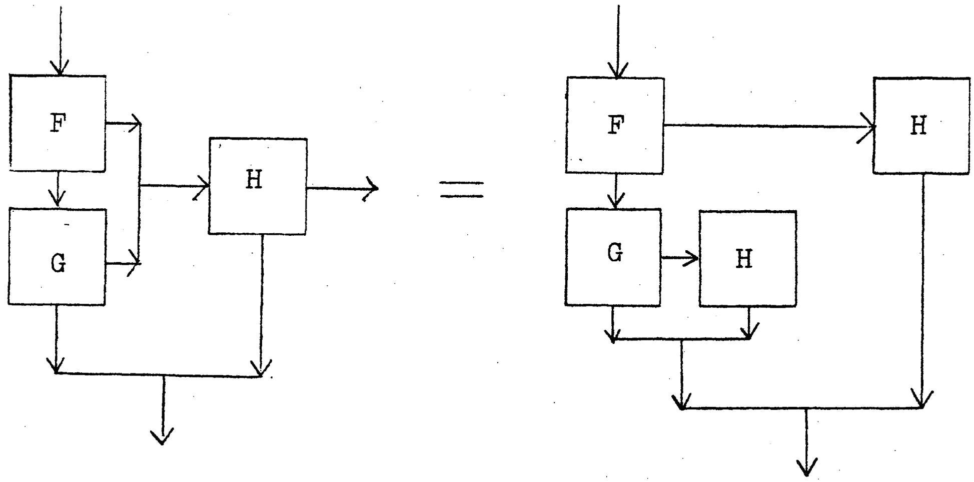

Figure 5 $FG+H=F(G+H)+H$ if $ef^{+}(H)$

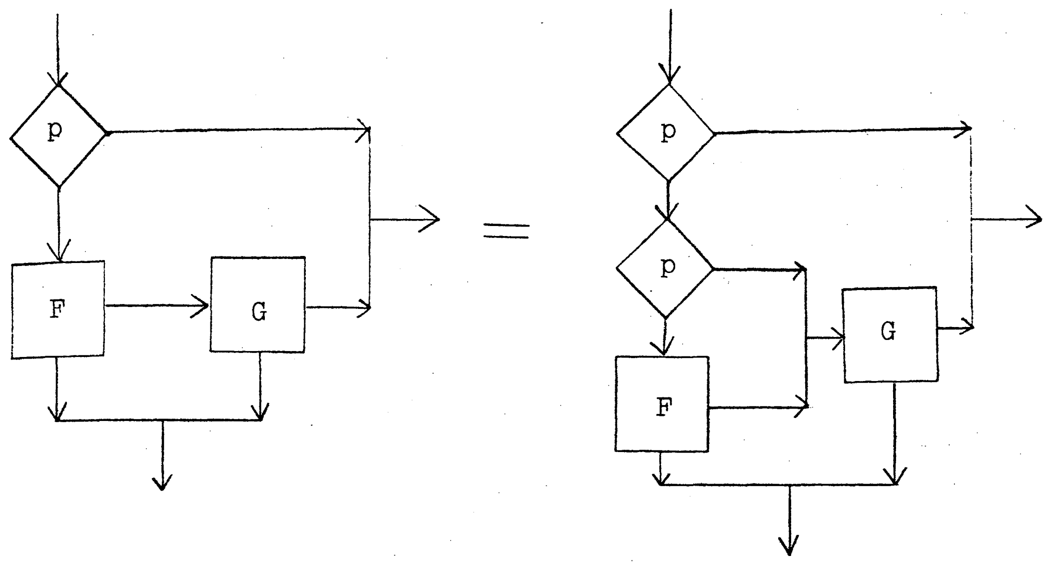

Figure 6 p(F+G)=p(pF+G)

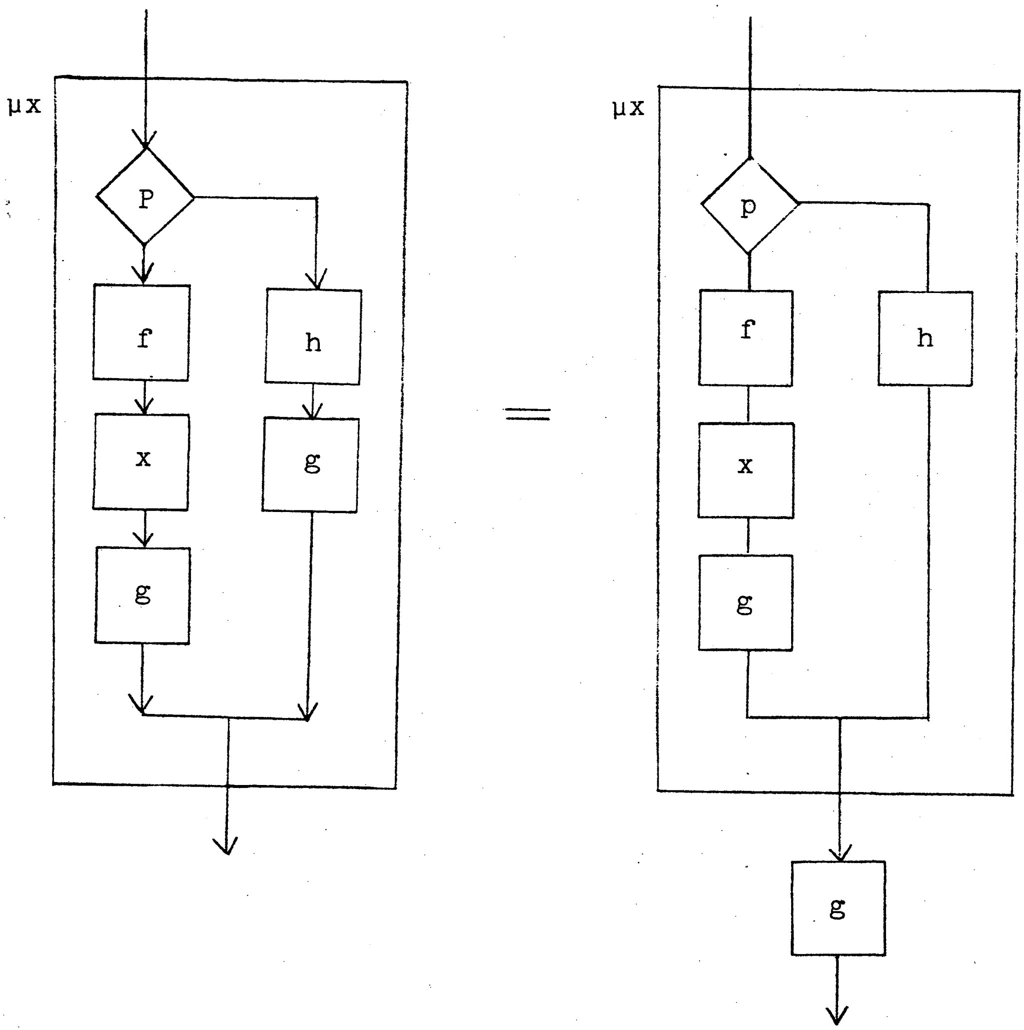

Figure 7 μx(pfxg+hg)=(μx(pfxg+h))g

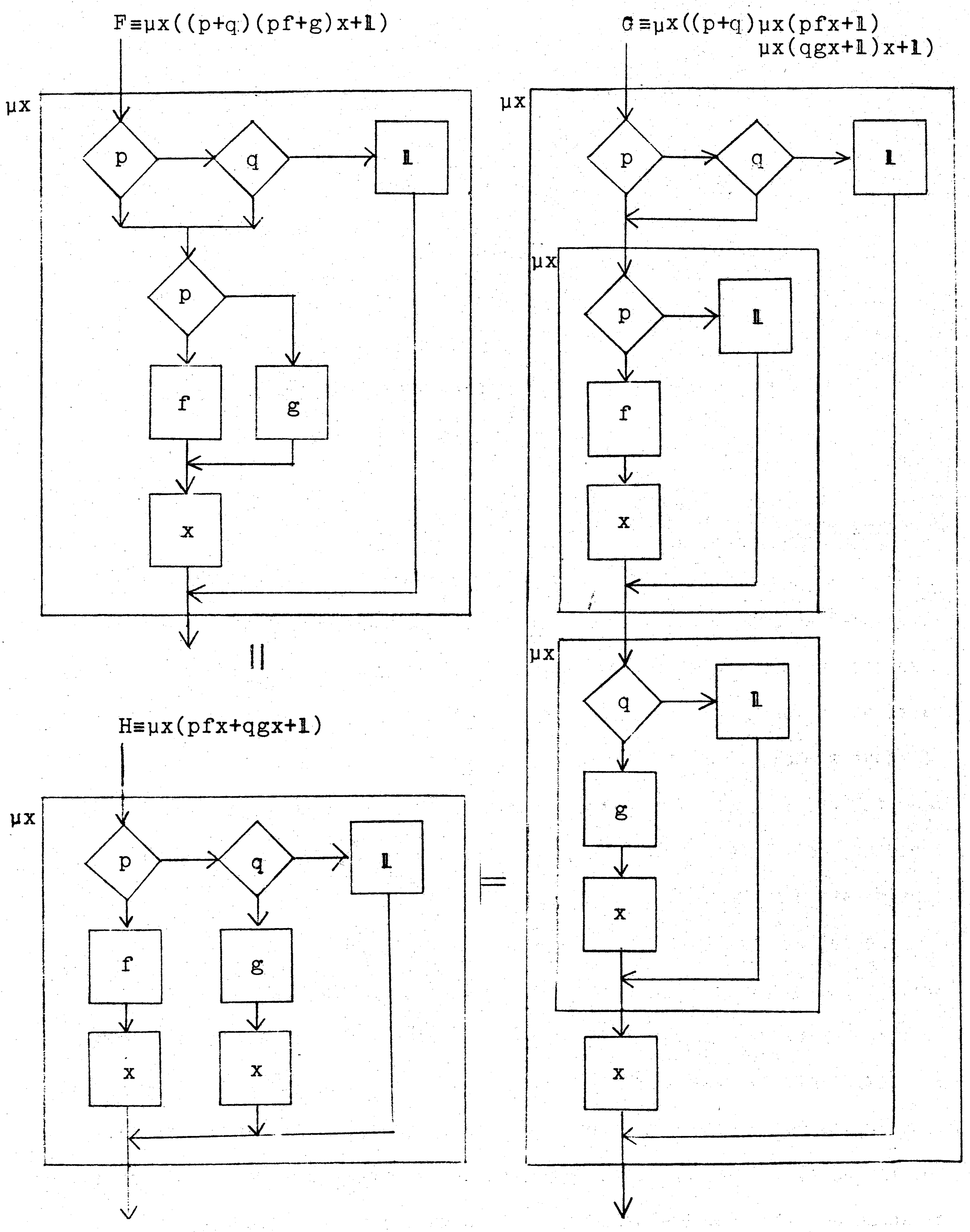

Figure 8 Example 4.3