ON THE COMPUTATIONAL COMPLEXITY OF PROGRAM SCHEMATA*

K. Weihrauch

TR 74-196

February 1974

Department of Computer Science
Cornell University
Ithaca, New York 14850

*This research has been supported in part by the National Science Foundation Grant GJ-33171X.

Abstract:

An ordering called "faster" is defined on the class of iterative program schemata. It is in good accordance with the intuition of "better" applied to program schemata. Many of the commonly used optimization techniques yield "faster" programs in this sense. For Ianov schemata the relation "faster" is decidable, but even on strong equivalence classes the ordering may be very complicated. For iterative program schemata there is a certain kind of speedup. Whereas there is an arbitrary slowdown for programs, slowdown for program schemata is limited.

ON THE COMPUTATIONAL COMPLEXITY OF PROGRAM SCHEMATA

Klaus Weihrauch

Introduction

Program schemata are studied in order to find general properties of programming languages. But although there seems to be an agreement about the necessity of investigating general questions of program optimization, especially the computational complexity of program schemata, until recently only very few papers deal with this or related topics. D.M. Symes and also N.A. Lynch present a generalized theory of computational complexity for computations with oracles. If the oracle is a function f, then the complexity of a computation depends on the program, on the input, and on f. The possibility that there may be several realizations of f with very different intrinsic computational complexities is not considered. R.L. Constable in his axiomatic approach takes the pair (f, tf) rather than f as a variable for the computational complexity of functionals, where tf is the complexity of f. Such an approach seems to be more reasonable for practical considerations. A.V. Aho and J.D. Ullman (see also N. Bracha) study optimization methods for loop-free program schemata, but they don't introduce

a precise concept of computational complexity. A.K. Chandra studies the lengths of computation sequences for a certain class of program schemata and proves a speedup theorem. As in the approach of Symes and Lynch the complexity of every function of the interpretation is fixed here.

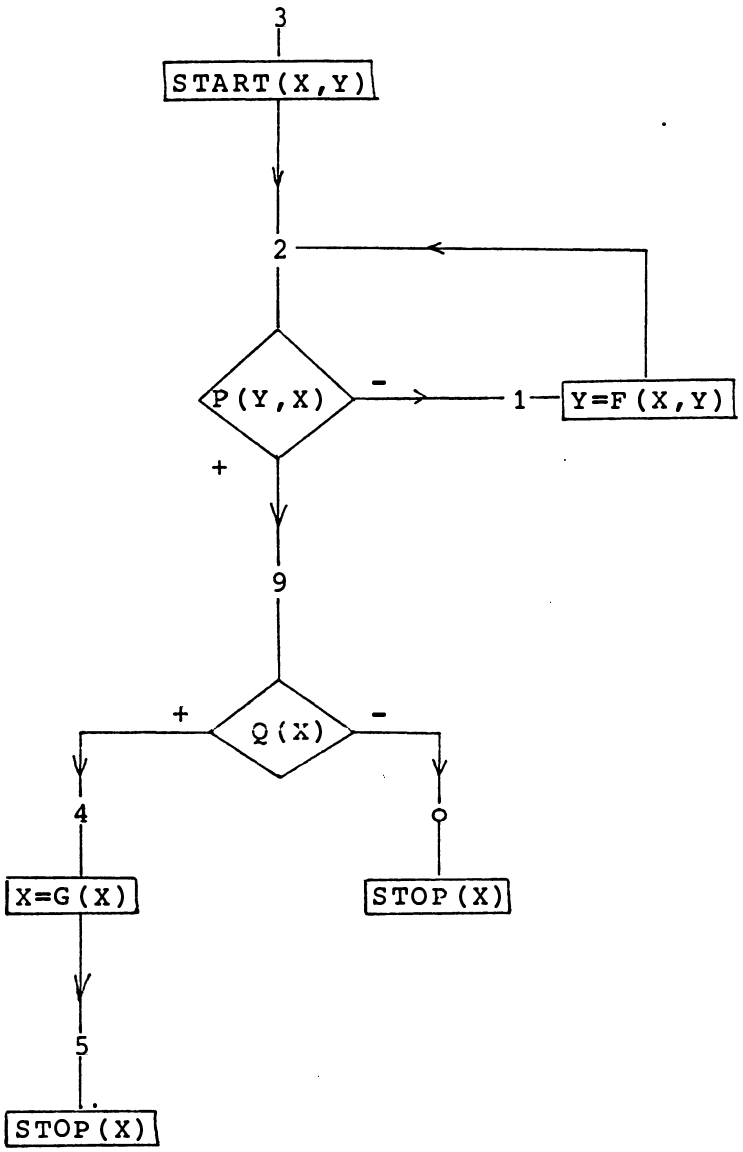
The greatest hindrance to the development of a theory of computational complexity of program schemata seemed to be the difficulty of finding reasonable basic definitions.

In this contribution we present a definition for the computational complexity of iterative program schemata. This definition will be justified informally. It induces a preordering on the set of program schemata. As applications we study this ordering for Ianov schemata and prove speedup and slowdown results for iterative program schemata.

Chapter 1: Basic Definitions and Properties

We assume that the reader already knows the one or the other definition for program schemata. Therefore, we shall only introduce notations we need in the subsequent chapters.

The following picture shows a program schema.



A program schema S can be considered to be a weighted directed graph. The nodes of this graph are the "states" of S. Let L be the set of states of S. For convenience we assume $L \subset \mathbb{N}$. The weights of the states are defined by a function π, where either

$\pi(i) \in \overline{B}_S$ = set of start -statements,

or $\pi(i) \in \overline{B}_F$ = set of assignment-statements,

or $\pi(i) \in \overline{B}_T$ = set of test-statements,

or $\pi(i) \in \overline{B}_E$ = set of stop-statements

The following conditions must hold for any $i \in L$.

If $\pi(i) \in \left\{ \begin{matrix} \overline{B}_S \\ \overline{B}_F \\ \overline{B}_T \\ \overline{B}_E \end{matrix} \right\}$ then i has $\left\{ \begin{matrix} 0 \\ 1 \\ 1 \\ 1 \end{matrix} \right\}$ predecessors and $\left\{ \begin{matrix} 1 \\ 1 \\ 2 \\ 0 \end{matrix} \right\}$ successors.

There is exactly one $l_o \in L$ with $\pi(l_o) \in \overline{B}_S$, called initial state. If $\pi(i_1) \in \overline{B}_E$ and $\pi(i_2) \in \overline{B}_E$ then $\pi(i_1) = \pi(i_2)$. For $i_1, i_2 \in L$ we write $i_1 \xrightarrow{S} i_2$, iff there is a directed path from i_1 to i_2. Furthermore, we assume that every simple variable or register name (X and Y in our example) is in the set $\overline{V} = \{V_i \mid i \in \mathbb{N}\}$, that every function variable (F and G in our example) is in the set $\overline{F} = \{F_i^{(j)} \mid i,j \in \mathbb{N}\}$, and that every test variable (P and Q in our example) is in the set $\overline{T} = \{T_i^{(j)} \mid i,j \in \mathbb{N}\}$, where the upper index denotes the number of arguments. Let PS be the set of all iterative program schemata.

Definition 1: Let D be a set and I be a mapping defined on $\overline{V} \cup \overline{F} \cup \overline{T}$ such that for all $i,j \in \mathbb{N}$:

$$I(V_i) \in D$$

$$I(F_i^{(j)}) : D^j \to D$$

$$I(T_i^{(j)}) : D^j \to \{+,-\}$$

Then I is called an interpretation. D is the domain of I.

We shall prove that, as for questions about equivalence, it is sufficient to consider Gödel-Herbrand interpretations, often called free interpretations, only.

Definition 2: An interpretation I is free, iff its domain D satisfies the following properties:

(1) D is the smallest subset A of $(\overline{V} \cup \overline{F})^*$ with

$\overline{V} \subseteq A$ and

if $d_1,\ldots,d_j \in A$ then $F_i^{(j)}(d_1,\ldots,d_j) \in A$,

(2) $I(V_i) = V_i$,

(3) $I(F_i^{(j)})(d_1,\ldots,d_j) = F_i^{(j)}(d_1,\ldots,d_j)$

It is useful to define every interpretation I also on the domain D_f of the free interpretations.

Definition 3: Let I be any interpretation. For any $F_i^{(j)}(d_1,\ldots,d_j) \in D_f$ we define induktively:

$$I(F_i^{(j)}(d_1,\ldots,d_j)) := I(F_i^{(j)})(I(d_1),\ldots,I(d_j))$$

For every program schema S and every interpretation I there is a single well defined computation. Let us define three sequences λ, ω, and ϱ, where λ describes the path through the schema, ω describes the subsequent register contents, and ϱ describes the operations performed in the single computation steps. Instead of a formal definition by induction we give an example.

Let S be the program schema shown at the beginning of this chapter. Let I be an interpretation with the following properties.

$D = \mathbb{N}$

$I(X) = 2,\ I(Y) = 8$

$F_I(u,v) = u+v$

$G_I(u) = u^2$

$P_I(u,v) = +<=>u = 1o$

$Q_I(u) = +<=>u = 2$

The following table shows the values of $\lambda(S,I)(k)$, $\omega(S,I)(k)$, and $\varsigma(S,I)(k)$ for $k \in \mathbb{N}$.

k	$\lambda(S,I)(k)$	$\omega(S,I)(k)$	$\varsigma(S,I)(k)$
0	2	(2,8)	*
1	1	(?,8)	$(P_I,(8,2))$
2	2	(2,1o)	$(F_I,(2,8))$
3	9	(2,1o)	$(P_I,(1o,2))$
4	4	(2,1o)	$(Q_I,2)$
5	5	(4,1o)	$(G_I,2)$
6	*	*	*
7	*	*	*
.	.	.	.
.	.	.	.
.	.	.	.

The definition of λ, ω, and ς in the general case should be clear from this example. Let us call $\lambda(S,I)$ the state sequence, $\omega(S,I)$ the value sequence, and $\varsigma(S,I)$ the complexity sequence or complexity of (S,I).

Lemma 1: For every interpretation I there is a free interpretation J such that for all program schemata $S \in PS$:

(1) $\lambda(S,I) = \lambda(S,J)$

(2) $\omega(S,I) = I \circ \omega(S,J)$

(3) $\varrho(S,I) = I \circ \varrho(S,J)$

J is uniquely defined by $J(T_j^{(n)})(d_1,\ldots,d_n) := I(T_j^{(n)})(I(d_1),\ldots,I(d_n))$.

The proof of this lemma is straightforward by induction on the steps of the computation.

Definition 4: Let S be a program schema and I be an interpretation. Then we define:

$|(S,I)| := \max\{k \mid \lambda(S,I)(k) \neq *\}$ = length of (S,I),

$ex(S,I) := \lambda(S,I)(|(S,I)|)$ = exit of (S,I),

val(S,I) : = content of the registers specified by the stop-statements after the computation stopped.

If $|(S,I)| := \infty$, then $ex(S,I) := val(S,I) := *$.

We now define strong equivalence and exit equivalence.

Definition 5: Let S_1, S_2 be program schemata.

Strong equivalence: $S_1 \equiv S_2 \iff$

$val(S_1,I) = val(S_2,I)$ for all interpretations.

Exit equivalence: $S_1 \overset{ex}{\equiv} S_2 \iff$

there is a permutation $\eta: \mathbb{N} \to \mathbb{N}$ such that

$ex(S_1,I) = \eta \circ ex(S_2,I)$ for all interpretations.

As an immediate consequence of Lemma 1 we have

Theorem 1: For $S_1, S_2 \in PS$ we have:

(1) $S_1 \equiv S_2 \iff val(S_1,I) = val(S_2,I)$ for all free interpretations,

(2) $S_1 \overset{ex}{\equiv} S_2 \iff ex(S_1,I) = \eta\, ex(S_2,I)$ for all free interpretations and some fixed permutation η.

Proof: $\Rightarrow$: immediate.

$\Leftarrow$: Let I be any interpretation and let J be the corresponding free interpretation by Lemma 1. Then

$val(S_1,I) = I \circ val(S_1,J) = I \circ val(S_2,J) = val(S_2,I)$.

The same argument holds in the case (2). Notice that $\eta I = I \eta$.

Q.E.D.

We have already called $\varrho(S,I)$ the complexity of (S,I). Complexities are useless if they cannot be possibly compared. Therefore, we introduce two pre-orderings on the set of complexities.

Definition 6: Let S_1, S_2 be program schemata, and let I_1, I_2 be interpretations. Then we define

$\varrho(S_1,I_1) \leq_0 \varrho(S_2,I_2) \iff |(S_1,I_1)| \leq |(S_2,I_2)|$

$\varrho(S_1,I_1) \leq_1 \varrho(S_2,I_2) \iff |(S_2,I_2)| = \infty$ or

$\varrho(S_1,I_1) = \varrho(S_2,I_2) \circ \delta$ for some injection $\delta : \mathbb{N} \to \mathbb{N}$.

A computation is simpler than another one in the first sense ($\leq_0$) if it is shorter. But we shall see that the length of the computation is not the only important property. Intuitively we would define: a schema S_1 is better than a schema S_2, iff S_1 is faster than S_2 on every computer.
A Computer interprets the input, function, and test variables and also defines a computation time for every single operation, i.e. a time assignment. Therefore, S_1 is better than S_2, iff S_1 is faster than S_2 for all interpretations and all computation time assignements. Then a single computation (S_1,I_1) is intuitively better than a computation (S_2,I_2) iff the first one is faster for all computation time assignements. But this relation is expressed exactly by our relation $\leq_1$.

Observation: The computation (S_1,I_1) is faster than the computation (S_2,I_2) for every computation time assignement, iff $\varphi(S_1,I_1) \leq_1 \varphi(S_2,I_2)$.

Proof: Suppose $\varphi(S_1,I_1) \leq_1 \varphi(S_2,I_2)$. Then every single operation appears at least as often in $\varphi(S_2,I_2)$ as in $\varphi(S_1,I_1)$. For every computation time assignement the first computation is therefore at least as fast as the second one. Now, suppose $\varphi(S_1,I_1) \not\leq_1 \varphi(S_2,I_2)$. Then there is a single operation α that appears more often in $\varphi(S_1,I_1)$ than in $\varphi(S_2,I_2)$. In this case we can choose a computation time

assignment, giving α a sufficient large time, such that the second computation is faster than the first one.

Q.E.D.

By this the sequences $\varrho(S,I)$ and the relation $\leq_1$ on them give us an adequate basis for studying the computational complexity of program schemata. The next definition is obvious.

Definition 7: Suppose $S_1, S_2 \in PS$. Then for i=o,1

$S_1 \leq_i S_2$: $\Leftrightarrow$ $\varrho(S_1,I) \leq_i \varrho(S_2,I)$ for all interpretations I,

$S_1 =_i S_2$: $\Leftrightarrow$ $S_1 \leq_i S_2$ and $S_2 \leq_i S_1$.

It is easy to show that $S_1 \leq_1 S_2 \Rightarrow S_1 \leq_o S_2$. The relations $\leq_i$ (i=o,1) are pre-orderings on the set of program schemata. Many of the commonly used optimization methods that work on the schema structure of the program yield faster programs in the sense $\leq_1$. It should be mentioned that the sequences $\varrho(S,I)$ contain all the information about the computation but no information about specific program data such as register names and states. Theorem 2 says that it is sufficient to consider only free interpretations.

Theorem 2: Suppose $S_1, S_2 \in PS$. Then for i=o,1

$S_1 \leq_i S_2 \Leftrightarrow \varrho(S_1,I) \leq_i \varrho(S_2,I)$ for all free interpretations.

Proof: =>: immediate

<= : Let I be any interpretation and J be the corresponding free interpretation by Lemma 1. Then $\varphi(S_1,I) = I \circ \varphi(S_1,J) = I \circ \varphi(S_2,J) = \varphi(S_2,I)$. (Similary for $\leq_o$)

Q.E.D.

The main difference between the classical theory of computational complexity and our new kind of complexity is that there are pairs of complexities S_1,S_2 incomparable under $\leq_1$.

We conclude this chapter with decidability results.

Theorem 3: The set $A = \{ (S_1,S_2) \mid S_1,S_2 \in PS, S_1 \not\leq_i S_2 \}$ is not recursive but recursively enumerable for i=o,1.

Proof: Suppose A is recursive. Choose S_1 such that S_1 diverges on all interpretations. But then $\{ S \mid S_1 \leq_i S \}$ = set of all program schemata that diverge on all interpretations would be decidable, which is false. Therefore A is not recursive. In order to show that A is recursively enumerable, enumerate all finite computations for free interpretations and all program schemata. $S_1 \not\leq_i S_2$ iff we find a free interpretation J such that $\varphi(S_1,J) \not\leq_i \varphi(S_2,J)$.

Q.E.D.

Chapter 2: The Computational Complexity of Ianov Schemata

In this chapter we prove the decidability of the relation $\leq_1$ on Ianov schemata. Furthermore we demonstrate that the relation $\leq_1$ may be very complicated even on equivalence classes of Ianov schemata. First we prove a normal form theorem that is true for all program schemata.

Definition 1: A program schema $S \in PS$ is in normal form, iff every $i \in L$ with $\pi(i) \in B_T$ has exactly one predecessor and $1_o \xrightarrow{S} i$ for every $i \in L$.

In other words, if S is in normal form, then every start or assignment statement has its individual subsequent test-tree. In our example S is not in normal form since 2 has 1 and 3 as predecessors. But we have the following theorem.

Theorem 1: For all $S \in PS$ there is $S' \in PS$ in normal form with $S \equiv S'$, $S \overset{ex}{=} S'$, and $\varrho(S,I) = \varrho(S',I)$ and $\omega(S,I) = \omega(S',I)$ for all I with $|(S,I)| < \infty$. By this especially $S =_i S'$ $(i=o,1)$.

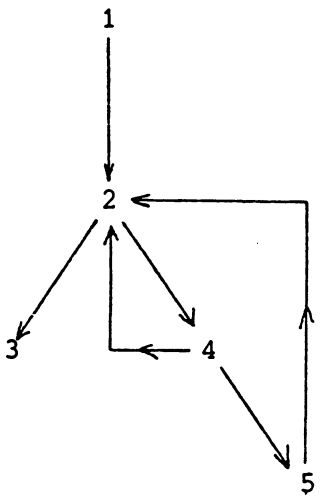
Proof: (outline) First cancel all $i \in L$ such that not $1_o \xrightarrow{S} i$. Add a cycling state $i_c \notin L$ with $\pi(i_c) \in \overline{B}_F$ and $i_c \to i_c$.

Then for every $i \in L$ with $\pi(i) \in \overline{B}_S \cup \overline{B}_F$ and $1_o \xrightarrow{S} i$ unwind the subsequent tests into an individual tree. After a certain depth the edges of this tree have entered nodes i with $\pi(i) \in \overline{B}_F \cup \overline{B}_E$, or it will be clear that they run into a loop for all interpretations. Connect these edges with the node i_c. The new schema S' has the desired properties.

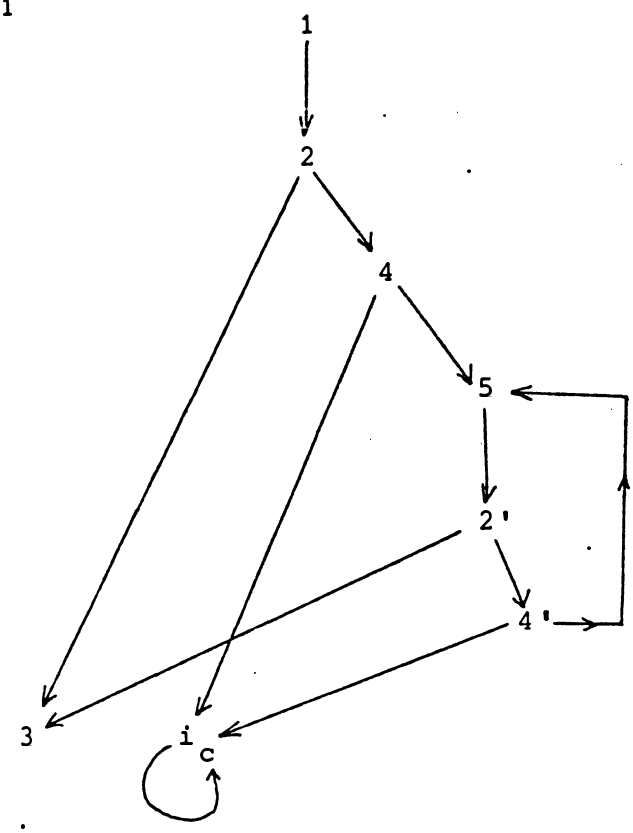
Q.E.D.

Example: Let us only symbolize the nodes of S and S'. Nodes with two outgoing edges symbolize tests.

S_1:



S_1':



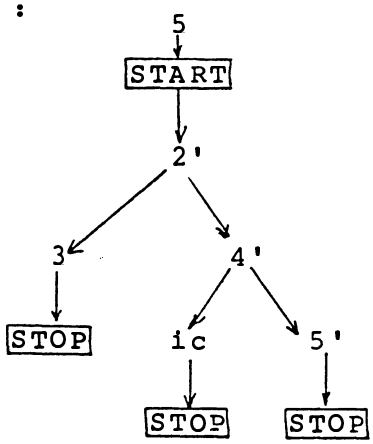
Definition 2: $S \in PS$ is a Ianov schema, iff S has only one register and only one argument function and test variables.

For Ianov schemata specification of input and output variables in the start and stop statements is unnecessary, furthermore statements are defined already by the function or test symbol. We shall use a simplified notation in this sense. The next definition has technical character.

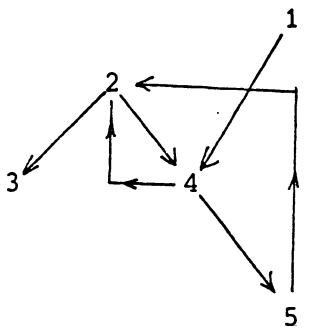
Definition 3: Suppose $S \in IS$, and S is in normal form. $Z := \{i \in L \mid \pi(i) \notin \overline{B}_T\}$ = set of super states. For every $i \in Z$ let $S^{(i)}$ be the Ianov schema that represents the test tree following state i. For every $i \in L - \{1_o\}$ let $S(i)$ be the schema obtained from S by going from 1_o directly to state i (here it is not necessary that S is in normal form).

Example: Let S_1, S_1' be the schemata of the previous example.

Then $S_1'^{(5)}$:



$S_1(4)$:



Theorem 2: The relation $\leq_1$ is decidable on IS.

Proof: Let S_1 and S_2 be Ianov schemata. Since the transformation into normal form is effective, we assume that S_1 and S_2 are in normal form. We shall construct a finite automaton A such that language (A) = $\emptyset$ iff $S_2 \leq_1 S_2$. But the property language (A) = $\emptyset$ is decidable on the set of finite automata. By Theorem 2 from Chapter 1 it is sufficient to consider only free interpretations. Let us now consider a sequence $\varsigma(S,I)$ for a free interpretation I and $S \in IS$. Since the arguments for the functions are well defined by the previous computation it is sufficient to write down the sequence of functions and tests, which we denote by $\bar{\varsigma}(S,I)$. Suppose

$$\bar{\varsigma}(S_1,I) = \tau_{1o}\, F_{i_1}\, \tau_{11}\, F_{i_2} \cdots \tau_{1,n-1}\, F_{i_n}\, \tau_{1n}$$
$$\bar{\varsigma}(S_2,I) = \tau_{2o}\, F_{j_1}\, \tau_{21}\, F_{j_2} \cdots \tau_{2,m-1}\, F_{i_m}\, \tau_{2m}\,,$$

where the τ_i are possibly empty sequences of test symbols. The construction of the automaton is based on the following property.

$$\bar{\varsigma}(S_1,I) \leq_1 \bar{\varsigma}(S_2,I) \iff (1),\ (2),\ \text{and}\ (3),$$

where

(1) $n \leq m$

(2) $F_{i_d} = F_{j_d}$ for $1 \leq d \leq n$

(3) $\tau_{1d} \leq_1 \tau_{2d}$ for $o < d < n$ (where $\tau_1 \leq_1 \tau_2$ iff τ_1 is a subsequence of a permutation of τ_2)

Construction of A:

Choose n such that $i \leq n$ for all test symbols T_i^1 occurring in S_1 or S_2. Define $C := \{0,1\}^{n+1}$. Every $c \in C$ assigns truth values to the T_i^1 for $0 < i < n$ by $c_i = 0$ iff T_i^1 is true. Let Z_1 and Z_2 be the sets of super states of S_1 and S_2. Define the automaton $A = (Z, C, \delta, z_o, z_e)$ as follows.

$Z := Z_1 \times Z_2 \cup \{ z_e \}$, $z_e \notin Z_1 \times Z_2$, is the set of states,
$C := \{ 0,1 \}^{n+1}$ is the input alphabet,
$z_o := (l_{o1}, l_{o2})$ is the initial state,
z_e is the accepting state, and
$\delta: Z \times C \to Z$ is defined by

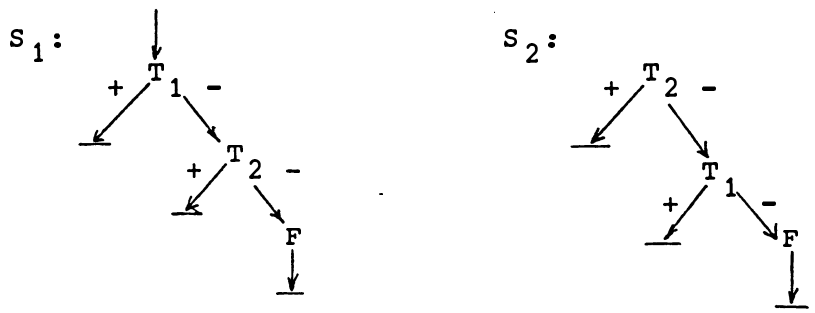
$$\delta((z_1,z_2),c) = \begin{cases} (ex(S_1^{(z_1)},c), ex(S_2^{(z_2)},c)) \text{ if } \pi_1(z_1) = \pi_2(z_2) \\ \qquad \text{and } \varrho(S_1^{(z_1)},c) \leq_1 \varrho(S_2^{(z_2)},c) \\ (z_1,z_2) \quad \text{if } \pi_1(z_1) \in \overline{B}_E \\ z_e \quad \text{otherwise} \end{cases}$$

$$\delta(z_e,c) = z_e.$$

Since $S_i^{(z_i)}$ contains only tests, the meaning of $ex(S_i^{(z_i)},c)$ and $\varrho(S_i^{(z_i)},c)$ is obvious. With the previous mentioned necessary and sufficient conditions it is easy to show that language $(A) = \emptyset \iff S_1 \leq_1 S_2$.

Q.E.D.

We shall now study the relation $\leq_1$ on strong equivalence classes of program schemata. The results give some limitations for optimization procedures. Suppose two schemata S_1 and S_2 symbolized by



Then we have $S_1 \equiv S_2$, S_1 and S_2 are both minimal under $\leq_1$ (i.e. $S \equiv S_i$ and $S \leq_1 S_i$ implies $S =_1 S_i$ for $i=1,2$), but neither $S_1 \leq_1 S_2$ nor $S_2 \leq_1 S_1$. We demonstrate the last property. Suppose I is such that $T_1(V) = +$, $T_2(V) = -$ then $\rho(S_2,I) \not\leq_1 \rho(S_1,I)$. Suppose I is such that $T_1(V) = -$ and $T_2(V) = +$ then $\rho(S_1,I) \not\leq_1 \rho(S_2,I)$. Therefore there may be incomparable elements in an equivalence class. Another observation has already been made in the proof of Theorem 2. Computations on strongly equivalent Ianov schema may only differ in the order of performing tests. First we propose a characterization of minimal Ianov schemata. For pre-orderings such as $\leq_1$ minimality is defined as follows.

Definition 4: Let $(A, \leq)$ be a pre-ordering. The element $x \in A$ is minimal iff $(\forall y \in A)\ y \leq x \Rightarrow x \leq y$.

Proposition: Let S be a Ianov schema in normal form. S is minimal in the equivalence class $\{ S' \in IS \mid S' \equiv S \}$, iff for every super state z of S the following holds: Take the test tree $S^{(z)}$. Let X be the set of end states of $S^{(z)}$. For every $x_1, x_2, \in X$ identify x_1 and x_2 iff $S(x_1) \equiv S(x_2)$ and $S(x_1) \leq_1 S(x_2)$. Repeat this process as long as possible. The final test tree is minimal under $\leq_1$.

By this proposition we have a method to optimize a Ianov schema S, i.e. to find $S' \leq_1 S$ with S' minimal.

We have already seen that there are pairs of minimal equivalent incomparable schemata. We shall prove a stronger result. The following Definition and Lemma prepare the proof.

Definition 5: Let $\{ o,1,2 \}$ be partially ordered by $\leq := \{ (o,2), (1,2), (o,o), (1,1), (2,2) \}$. Let $R \subset \{ 0,1,2 \}^{\mathbb{N}}$ be the set of all ultimately periodic sequences $((\forall r \in R)(\exists\, m,n \in \mathbb{N})$ $x \geq a \Rightarrow r(x+m) = r(x))$. Extend $\leq$ to R by

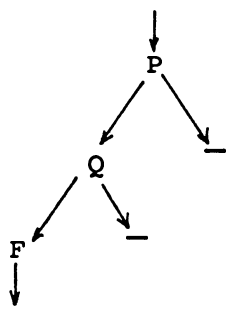
$$r_1 \leq r_2 \iff (\forall i \in \mathbb{N})\ r_1(i) \leq r_2(i).$$

By this $(R, \leq)$ is a partial order.

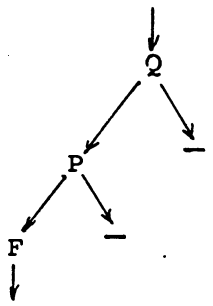
Lemma 1: There is $S \in IS$ and a mapping $\alpha: R \to K_S = \{ S' \in IS \mid S' \equiv S \}$ with $r_1 \leq r_2 \iff \alpha(r_1) \leq_1 \alpha(r_2)$ for all $r_1, r_2 \in R$.

Proof: Let $\rightarrow C_o \rightarrow$, $\rightarrow C_1 \rightarrow$, and $\rightarrow C_2 \rightarrow$ represent the following parts of Ianov schemata.

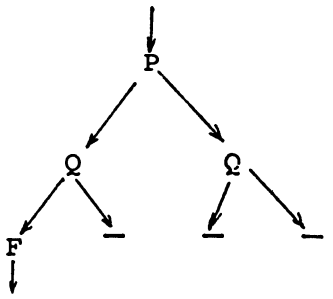
$\rightarrow C_o \rightarrow$:



$\rightarrow C_1 \rightarrow$:



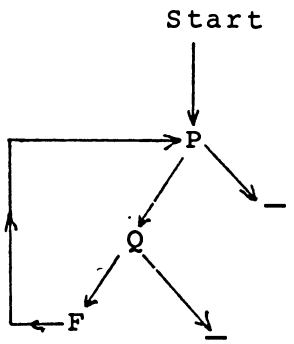
$\rightarrow C_2 \rightarrow$:



(Here — stands for the Stop statement.)

If we complete these three parts into schemata C_o', C_1', C_2', then $C_o' \equiv C_1' \equiv C_2'$ and the ordering under $\leq_1$ is isomorphic to the ordering $(\{o,1,2\}, \leq)$ in Definition 5. The schema S and the mapping α are now easily defined.

S:



Suppose $r \in R$. Let m be the smallest ultimate period and a be the smallest number such that $r = r_1 r_2 \ldots r_a (r_{a+1} \ldots r_{a+m})^\omega$.

$\alpha(r) :=$ Start $\rightarrow C_{r_1} \rightarrow C_{r_2} \rightarrow \ldots \rightarrow C_{r_a} \rightarrow C_{r_{a+1}} \rightarrow \ldots \rightarrow C_{r_{a+m}}$

The property $r \leq s \iff \alpha(r) \leq_1 \alpha(s)$ can now easily be shown.

Q.E.D.

The ordering $(R, \leq)$ is very complicated:

Lemma 2: $(R, \leq)$ contains dense chains and an infinite number of pairwise incomparable elements.

Proof: Define $r << r'$ iff $r(i) \leq r'(i)$ and $r'(i) \not\leq r(i)$ for infinitely many i and $r \leq r'$. Suppose $r << r'$. Then there is r" with $r << r'' << r'$, and $(R, \leq)$ contains dense chains. On the other hand all the elements in $R \cap \{0,1\}^{\mathbb{N}}$ are pairwise incomparable.

Q.E.D.

Theorem 3: There is a strong equivalence class of Ianov schemata that contains dense chains and an infinite number of pairwise incomparable minimal elements under $\leq_1$.

Proof: (immediate by Lemma 1 and Lemma 2)

Q.E.D.

We have the following situation. By the proposition every Ianov schema can be optimized into a minimal one, but by Theorem 3 there may be several minimal ones. In our model of computational complexity we have no criterion to select

a unique one among them. Theorem 3 also says that step by step optimization could run into an infinite chain never obtaining an optimal schema. But this could happen only to a very careless optimizer. In this connection Theorem 4 says that a descending chain does not need to have a lower bound.

Theorem 4: There is a strong equivalence class of Ianov schemata that contains a descending chain having no lower bound.

Proof: Define $r_n \varepsilon R$ by $r_n = o1o1^2o1^3o\ldots o1^n 2^\omega$.
The schemata $\alpha(r_o)$, $\alpha(r_1)$, ... form a descending sequence under $\lesssim_1$ that has no lower bound in IS.

Q.E.D.

Chapter 3: Speedup and Slowdown

One of the most impressive results in the theory of computational complexity is the speedup theorem. For every recursive function r there is a function f such that for every program i for f there is an r-better program. But generally there is no method to find the better and better programs. It is also a simple fact, that any

program can be made slower by an arbitrary amount. Are there similar properties for program schemata? The following theorems will give some answers. First Ianov schemata are considered.

Theorem 1: For every Ianov schema S there is a Ianov schema S' with $S \equiv S'$, $S' \leq_1 S$ and S' is minimal.

Proof: By the proposition of Chapter 2.

Therefore, there is no speedup in the class of Ianov schemata. The next theorem gives a limit for the slowdown.

Theorem 2: Suppose $S_1, S_2 \in IS$, $S_1 \equiv S_2$. Then there is a constant $d \in \mathbb{N}$ with $(\forall I)|(S_1,I)| \leq d(|(S_2,I)|+1)$

Proof: By Chapter 1 it is sufficient to consider only free interpretations. For any free interpretation I we have $val(S_1,I) = val(S_2,I) = F^1_{i_k} F^1_{i_{k-1}} \ldots F^1_{i_1} V_o$. For S_1 there is $a \in \mathbb{N}$ such that at most a tests are to be performed between two function applications. Therefore, $|(S_1,I)| < (a+1)|(S_2,I)| + a$.

Q.E.D.

For Ianov schemata slowdown is at most linear. It is easy to show that linear slowdown is actually possible.

Theorem 3: Suppose $S \in IS$ and $d \in \mathbb{N}_+$. Then there is $S' \in PS$, $S' \equiv S$ with $(\forall I)|(S',I)| > d(|(S,I)|+1)$ and $S \leq_1 S'$.

Proof: Suppose S has a states. Expand S by inserting a dummy test tree of depth d between every pair of states.

Q.E.D.

There are Ianov schemata that can be optimized by a linear amount (Theorem 3), but no Ianov schema can be optimized by more than a linear amount. A similar, but somewhat weaker property can be proved for iterative program schemata. The following theorem corresponds to Theorem 2.

Theorem 4: Suppose $S, S' \in PS$, $S \equiv S'$. Then there is a polynominal p such that

$$(\forall I)(\exists J) \quad |(S,I)| = |(S,J)| \text{ and } |(S',J)| \leq p(|(S,J)|)$$

In other words, on a representative subset of interpretations every slowdown is at most polynominal.

Proof: Suppose $|(S,I)| = n < \infty$. Since at most $n+m$ values of the domain D_I of I can be used in this computation (here m is the number of registers in S), there is a finite interpretation J with $|D_J| = n+m$ and $|(S,J)| = n$. Since $S \equiv S'$ we know that $|(S',J)| < \infty$. Suppose that S' has l states and k registers. If $|(S',J)| > (n+m)^k \cdot l$, then one instantaneous situation, defined by the state and the register contents has appeared twice. In this case the computation must cycle. Therefore $|(S',J)| \leq (n+m)^k \cdot l$.

Q.E.D.

Theorem 4 can be interpreted as follows. Suppose $f: \mathbb{N} \to \mathbb{N}$ is increasing and $(\forall I)\ |(S',I)| \geq f(|(S,I)|)$. Then f is bounded by some polynominal p. A "uniform" slowdown is at most polynominal. In analogy to Theorem 3 we show now, that every polynominal slowdown can actually be obtained.

Theorem 5: For every $S \in PS$ and every polynominal p there is $S' \in PS$ such that $S \leq_1 S'$ and $(\forall I)\ |(S',I)| \geq p(|(S,I)|)$. S' can be constructed from S.

Proof: It is sufficient to prove the Theorem for the cases $p(x) = x+1$ and $p(x) = x^2$. The case $p(x) = x+1$ is trivial. Suppose $p(x) = x^2$. Take S, store input values on special new registers; interrupt the schema at every state and insert a copy of S that operates on individual registers, not disturbing the computation of the main program. The resulting schema has the desired properties.

Q.E.D.

The reader may have observed the weak formulation of Theorem 4 compared to that of Theorem 2. The reason is the possibility of arbitrary slowdown on a certain subset of interpretations.

Theorem 6: [+)] For all $S \in PS$ and all recursive functions $r: \mathbb{N} \to \mathbb{N}$ there is $S' \in PS$, $S' \equiv S$, $S \leq_1 S'$, such that

$$(\forall I)(\exists J) \quad \varphi(S,I) = \varphi(S,J) \text{ and}$$

$$\varphi(S,J) \leq_1 \varphi(S',J) \text{ and}$$

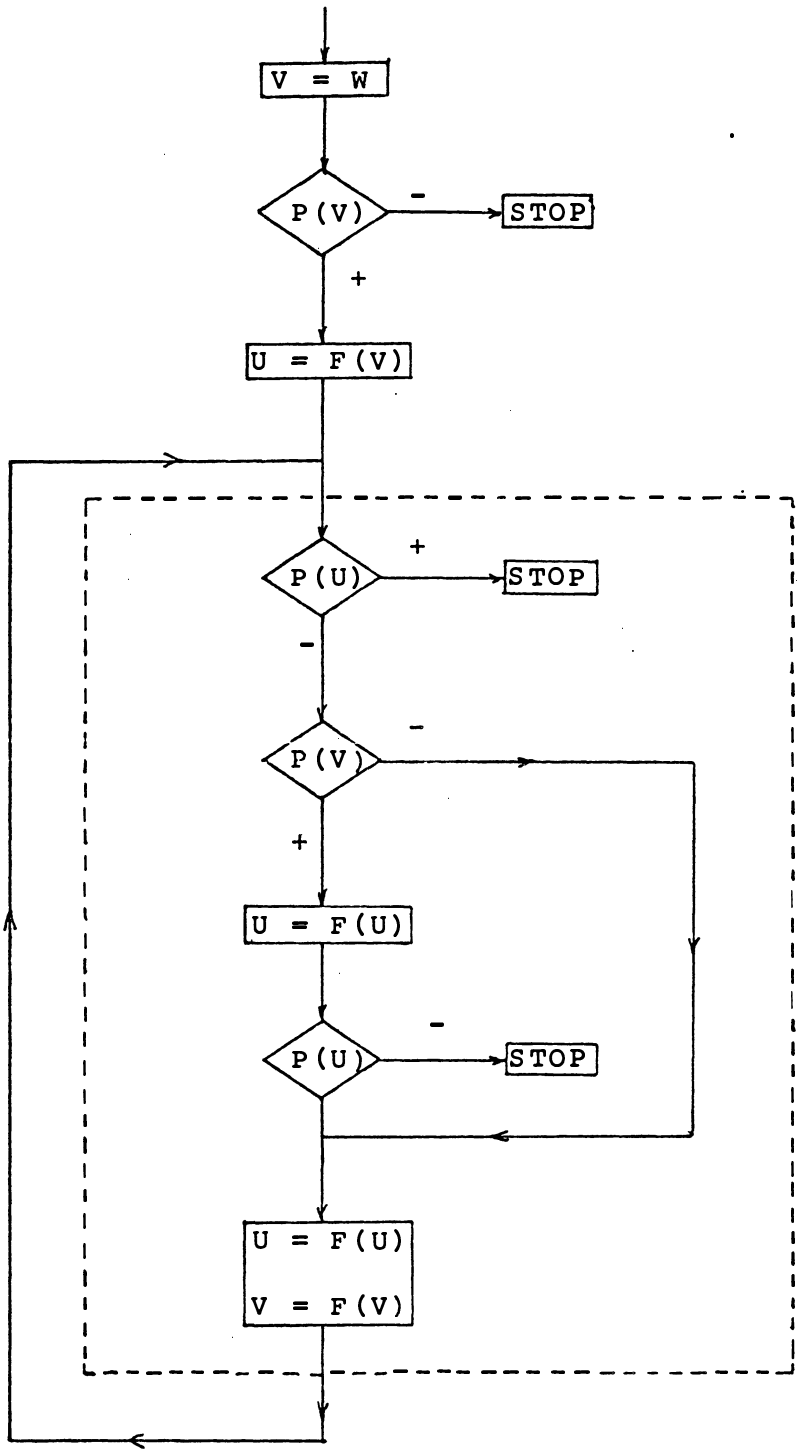
$$r(|(S,J)|) \leq |(S',J)|.$$

Proof: (outline) The idea is to simulate the computation of some fixed program for r on the domain of J, D_J. This simulation works only for interpretations J that satisfy certain sufficient conditions. During the simulation we test from time to time whether J satisfies these conditions, if not, the simulation is stopped. The following program schema S diverges for an interpretation J, iff the subsequent values of $P_J F_J^i (W_J)$ for i=o,1,2,... are +-+--+---+----+ ... .

Suppose, the symbols P and F do not appear in the schema S. Let M be the set $\{ F_J^i (W_J) \mid i \in \mathbb{N} \}$. Two values $u,v \in M$ are said to be similar, iff $P_J F_J^k u = P_J F_J^k v$ for all $k \in \mathbb{N}$. We choose a program for r on a machine with a finite number of registers to store integers as its storage, that has the functions $x \leftarrow 0$ and $x \leftarrow y+1$ and the test $x=y$ as basic operations. We simulate every $i \in \mathbb{N}$ by $F_J^i (W_J)$. But we have to make sure, that all $F_J^j (W_J)$ used in the simulation are pairwise unsimilar.

+) This theorem has also been proved by E. Ashcroft and K. Mehlhorn independently.

S_o:



This can be done by executing the computation of S_o for a sufficient number of steps. If the part in dotted lines has been executed 2n times, then we are sure that at least the n values $F_J^j(W_J)$ $(o \leq j \leq n-1)$ of M are pairwise unsimilar. The new schema S' can be constructed by the following instructions.

(1) Execute schema S. If S diverges, then S' diverges. If S stops, store the output. This output will be the output of S', whenever it stops.

(2) Compute $u \in M$ such that $u = F_J^n(W_J)$ where $n = |(S,J)|$. For this purpose execute S and S_o in a parallel way such that for every step of S the dotted box of S_o is executed one time. If there is a stop in the simulation of S_o, then S' stops. After S has stopped, V contains u.

(3) Make sure that all the values $F_J^i(W_J)$ with $o \leq i \leq n$ are pairwise unsimilar. Execute once more S and S_o in a parallel way, but now with two executions of the dotted box for every step of S. If there is a stop in the simulation of S_o before S stops, then S' stops. When S stops, then interrupt the simulation of S_o.

(4) Simulate the program for r.

(a) Simulation of $x \leftarrow O$: $X = W$,

(b) simulation of the test x=y: test whether PF^kX and PF^kY have the same truth values for k=o,1,...,m, where m is the smallest number such that for two numbers k_1, k_2 with $O \leq k_1 < k_2 \leq m$ the two tests have the values +,

(c) simulation of $x \leftarrow y+1$: $X=F(Y)$, in addition try to extend the domain for the simulation of r by continuing the computation of S_o interrupted in (3) by two executions of the dotted box. If we have a stop here, then J is not good and S' stops. Otherwise go on with the simulation of r.

Since r is total recursive this process will stop in any case. This ends the description of schema S'. If J behaves as I on the values needed for the schema S, and if $|(S_o,J)| = \infty$, then the simulation of r will be fully executed. In this case we have $|(S',J)| \geq r(|(S,J)|)$. This follows from the fact that $n+t(n) > r(n)$, where $t(n)$ is the number of steps to compute $r(n)$. The properties $S' \equiv S$, $S \leq_1 S'$, $\varphi(S,I) = \varphi(S,J)$, and $\varphi(S,J) \leq_1 \varphi(S',J)$ are obvious.

Q.E.D.

It should be mentioned that the introduction of a new function symbol F and a new test symbol P is not necessary, whenever S contains at least one function and one test symbol. The simulation of r can then be performed on that part of the domain of J, that has not been used for the computation (S,J).

Although for two equivalent program schemata the lengths of computations are not necessarily in polynominal relation for all interpretations, the relation is recursive in any case.

Theorem 7: For all $S_1, S_2 \in PS$ with $S_1 \equiv S_2$ there is a (total) recursive function $c: \mathbb{N} \to \mathbb{N}$ such that

$$(\forall n)\ (\forall I)\ |(S_1,I)| \leq n \Rightarrow |(S_2,I)| \leq c(n)$$

A program for c can be computed from S_1 and S_2.

Proof: Unwind S_2 into a tree taking into account only those paths defined by an interpretation I with $|(S_1,I)| \leq n$. This process must stop at some depth m. Otherwise there were arbitrarily long paths and, by Königs Lemma, an infinite path, therefore an I with $|(S_1,I)| \leq n$ and $|(S_2,I)| = \infty$, which contradicts $S_1 \equiv S_2$. Define $c(n) := m$.

Q.E.D.

The previous theorems give limitations for the efficiency of certain program optimization methods, namely those methods that are only dependent on the structure of the program and not on the interpretation of the symbols. Is there an optimal schema S' for every schema S such that $S' \leq_1 S$, $S' \equiv S$? For Ianov schemata the answer is yes. We shall show now that the answer is no for the class of iterative schemata. There is $S_o \in PS$ such that for all $S \in PS$, $S \equiv S_o$, there is a better schema $S' \equiv S_o$ with $S' <_o S$ and $S' <_1 S$. The better schema can effectively be constructed. The method is given by Theorem 8.

Theorem 8: Suppose $S \in PS$ and $w = F(t_1,t_2,\ldots t_j) \in D_f$ (= domain of the free interpretations). Then there is $S' \in PS$, $S' \equiv S$, such that for all free interpretations I the sequence $\varphi(S',I)$ is obtained from $\varphi(S,I)$ by cancelling all terms $(F,(t_1,\ldots,t_j))$ up to the first one. S' can be effectively constructed from S and w.

Proof: (outline) First, we observe that there is only one way to compute w from the input values. Let us call the intermediate values "subvalues". The set of subvalues of w is finite. Let us now handle S by a supervisor. The supervisor can store which of the subvalues of w, or w, or none of them is in which register of S. Therefore it can also recognize the instances when w is computed and the instances when w is requested. So the supervisor can store w at the first instance when it is computed in a special new register, can avoid every new computation of w and, if requested anywhere, supply the schema S with w. For doing all this, the supervisor needs only a finite memory. But a finite memory can be added to S by adding states. So S' can be constructed by straightforward programming.

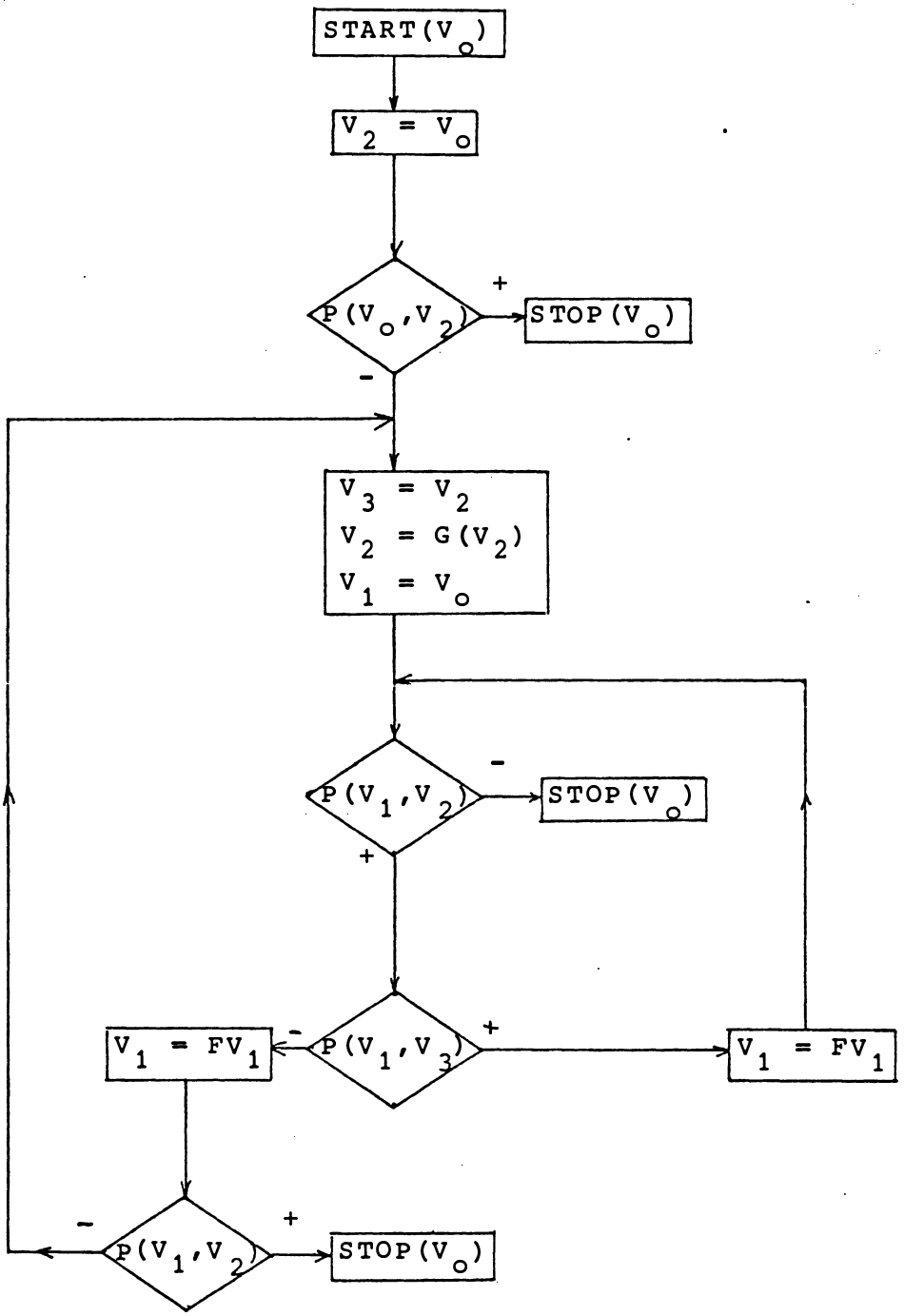
Q.E.D.

Theorem 9: (speedup) $(\exists S_o \in PS)(\forall S \in PS, S \equiv S_o)(\exists S' \in PS, S' \equiv S_o, S' <_1 S)$ $(\exists r: \mathbb{N} \to \mathbb{N}$, recursive, nondecreasing, unbounded)

$$(\forall I)\ |(S',I)| \leq (1-r)\,(|(S,I)|)$$

Furthermore S' and a program for r can effectively be found.

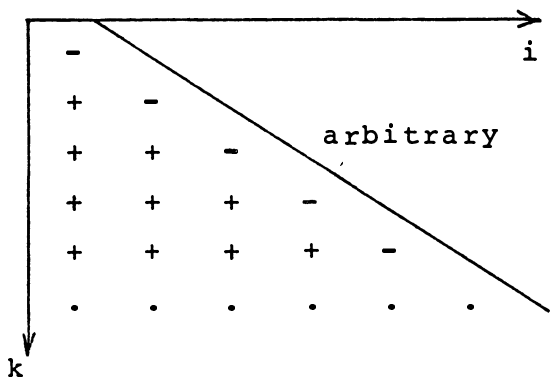
Proof: Let S_o be defined by the following diagram.



It is sufficient to consider only free interpretations.

S_o is constructed such that for all free I schema S_o diverges iff

$$\delta_I^{ik} := P_I(F^i(V_o), G^k(V_o)) = \begin{cases} - & \text{for } i = k \\ + & \text{for } i < k \\ \text{arbitrary} & \text{otherwise} \end{cases}$$



Set $\gamma_I^{ik} := (P_I,(F^i(V_o), G^k(V_o)))$.

Since each of the values $F^i(V_o)$ is used the more often the longer a computation is, schema S_o can be improved by Theorem 8 arbitrarily often. We shall show now, that every $S \equiv S_o$ can be improved for a similar reason.

Assertion: $(\forall S, S \equiv S_o)\ (\exists d: \mathbb{N} \to \mathbb{N}, d \text{ recursive})\ (\forall I, \text{free})\ (\forall n)\ (\forall i, i \leq n)\ (\exists k, n < k \leq d(n))\ |(S,I)| > d(n) \Rightarrow \gamma_I^{in} = \varphi(S,I)(k)$.

Proof of the assertion: There is $m \in \mathbb{N}$ such that $(\forall I, \text{free})$ $|(S_o,I)| > m \Rightarrow \delta_I^{ik} = +$ and $\delta_I^{kk} = -$ for all $o \leq i < k \leq n$. There are $a, b \in \mathbb{N}$ such that $m = an^2+b$ is an appropriate choice. Define d by $d(n) := c(m) = c(an^2+b)$. Here c is the function from Theorem 7 with the property $(\forall l)\ (\forall I)\ |(S_o,I)| \leq l \Rightarrow |(S,I)| \leq c(l)$. We can assume $(\forall l)\ c(l) \geq l$. Suppose to the contrary that there is a free

interpretation I with $|(S,I)| > d(n)$ but for some j, $0 \leq j \leq n$, $\gamma_I^{jn} \neq \varphi(S,I)(k)$ for all $n < k \leq d(n)$. Certainly $\gamma_I^{jn} \neq \varphi(S,I)(k)$ for all $k \leq n$, since $G^n(V_o)$ has to be computed first which requires n steps. Therefore $\gamma_I^{jn} \neq \varphi(S,I)(k)$ for all $k \leq d(n)$. Now define another free interpretation J such that J coincides with I up to possibly the value δ_J^{jn}. Define $\delta_J^{jn} = -$ if $j < n$, $\delta_J^{jn} = +$ if $j=n$. But then $\varphi(S,I)(k) = \varphi(S,J)(k)$ for all $k \leq d(n)$. We deduce $|(S,J)| > d(n)$. On the other hand by the definition of δ_J^{jn} we are sure that $|(S_o,J)| \leq m$, and $|(S,J)| \leq c(m) = d(n)$ by Theorem 7, which is a contradiction. This proves the assertion.

We conclude: For any n, if $|(S,I)| > d(n)$ then all the values V_o, $F(V_o)$, ..., $F^n(V_o)$ are used as arguments of P_I between step n and step d(n). Suppose S has q registers. Then for every $i \in \mathbb{N}$ the values V_o, $F(V_o)$, ..., $F^q(V_o)$ are used as arguments between step $d^i(q)$ and step $d^{i+1}(q)$ if $|(S,I)| = k$ and $d^{i+1}(q) \leq k$. (Remember that $(\forall l)\ d(l) > c(l) \geq l$). Therefore in each of these intervals at least one of these values has to be re-evaluated. Application of Theorem 8 to S and the values V_o, $F(V_o)$, ..., $F^q(V_o)$ (q+1) - times gives a faster schema S'. If $|(S,I)| \geq d^j(q)$ then $|(S',I)| + j \leq |(S,I)|$ for every interpretation I. Define r by $r(n) := \max\{j \mid d^j(q) \leq n\}$ (define $\max(\emptyset) = 0$). Since d is recursive

and $(\forall l)\ d(l) > 1$ the function r is recursive, nondecreasing and unbounded. $|(S,I)| = n$ implies $|(S,I)| = n \geq d^{r(n)}(q)$, therefore $|(S',I)| + r(n) \leq |(S,I)|$.

Q.E.D.

Since we don't know very much about the schema S, we don't know very much about the function r. By Theorem 6 we know that the function c in Theorem 7 may be very much increasing, so r may be a very slowly increasing function. But from Theorem 5 we can conclude that for some interpretations our result is not too bad.

Corollary:

$(\exists S_o \in PS)\ (\forall S \in PS,\ S \equiv S_o)\ (\exists S' \in PS,\ S' \equiv S_o,\ S' \leq_1 S)\ (\forall I)\ (\exists J)$

(1) $|(S',J)| \leq (1 - \frac{1}{a} \log \log)\ (|(S,J)|)$ for some $a \in \mathbb{N}$,

(2) $|(S,J)| \leq p(|(S_o,J)|)$ for some polynominal p,

(3) $|(S_o,J)| = |(S_o,I)|$.

Proof: By Theorem 4 we know that (2) and (3) hold. Suppose $p(n) \simeq n^k$. For the interpretation J we can substitute in the proof of Theorem 9 c(n) by n^k and d(n) by n^{2k}. But then $r(n) \simeq \frac{1}{a} \log \log n$ for some a.

Q.E.D.

A.K. Chandra proved a stronger speedup theorem not regarding the condition $S' \leq_1 S$. He showed: $(\exists S_o)(\forall S \equiv S_o)(\exists S' \equiv S_o)(\exists m \in \mathbb{N})$ $(\forall I) \quad |(S',I)| \leq |(S,I)|^{1-\frac{1}{m}}$. But there is no obvious way to generalize this result such that also $S' \leq_1 S$ holds.

Let us conclude with a last remark. One might suspect that whenever the function c of Theorem 7 cannot be bounded by a polynominal, then there is an S_2', $S_2' \equiv S_2$, $S_2' \leq_1 S_2$, such that the polynominal relation holds for S_1 and S_2'. This is not always true.

Theorem 10:

$(\exists S_o)$ $(\forall r: \mathbb{N} \to \mathbb{N}$, recursive$)$ $(\exists S \equiv S_o)$ $(\forall S' \equiv S_o, S' \leq_1 S)(\forall f: \mathbb{N} \to \mathbb{N})$ $[\ (\forall I)|\ (S',I)| \leq f(|(S_o,I)|)\ ] \Rightarrow f(n) \geq r(n)$ for almost all n.

Proof: Use two incomparable minimal schemata, the construction of the proof for Theorem 6 and the compression theorem from the classical complexity theory.

Q.E.D.

The condition $\leq_1$ for optimizing may make considerable improvement of the lengths of computation impossible.

Acknowledgement:

A portion of this work was performed while the author visited the Cornell University Computer Science Department from July to October 1973. The author wishes to thank Kurt Mehlhorn and Ray Titelbaum for helpful discussions.

References:

A.V. Aho, J.D. Ullman,

Transformations on straight line programs,
2 nd Annual ACM Symposium on Theory of Computing (197o),
136 - 148

N. Bracha,

Transformations on loop-free program schemata,
Department of Computer Science, University of Urbana, Illinois,
Report No. UIUCDCS - R - 72 - 516 (1972),

A.K. Chandra,

Efficient compilation of linear recursive programs,
14 th Annual Symposium on Switching and Automata Theory (1973),
16 - 25

R.L. Constable,

Type two computational complexity
5 th Annual ACM Symposium on Theory of Computing (1973),
1o8 - 121

N.A. Lynch,

Relativization of the theory of computational complexity,
Massachusetts Institute of Technology, Project MAC, TR-99 (1972,

D.M. Symes,

The extension of machine-independent computational complexity
theory to oracle machine computation and to the computation
of finite functions,
Thesis, University of Waterloo, Waterloo, Ontario (1971),