Acta Informatica 2, 311—333 (1973)

On Procedures as Open Subroutines. I

Hans Langmaack

Received June 5, 1973

Summary. An ALGOL program with open subroutines or macro program is a program whose procedures may be implemented as open subroutines. A macro program may be considered to be an abbreviated notation of a program without procedures. It is proved that the so called macro program problem is algorithmically unsolvable for ALGOL 60 and other ALGOL-like languages: There does not exist any algorithm which decides for any given program whether it is a macro program or not (Theorem 4.3 and 4.4 in part II). Sublanguages of ALGOL-like languages for which the macro program problem is solvable are furtheron investigated (Theorem 4.1, 5.2–5.5 in part II). For this purpose macro grammars theory is applied.

The methods of the paper are developped in part I and the key lies in a generalized langugage ALGOL 60-P-G which has the so called modularity property: There is an effective process which constructs for every original ALGOL 60-P-G program a formally equivalent one without procedure nesting (Theorem 3.3). The process mainly works by eliminating global procedure parameters (Theorem 3.1). ALGOL 60 and other known ALGOL-like languages do not have the modularity property (Theorem 3.4). Elimination of global procedure parameters is successful only for sublanguages of ALGOL 60 (Theorem 2.3–2.7).

1. Introduction

Mainly two different techniques to implement procedures in higher programming languages are available, the closed subroutine technique and the open subroutine or macro technique. The first technique is more general than the latter and creates statically shorter machine programs, but the macro technique yields shorter execution times if this technique is applicable. As there exist higher language programs, especially systems programs, for which extremely short execution times are desired we must be interested in criteria which allow to decide whether the macro technique can be applied. We introduce the notion of a program with open subroutines or macro program (Definition 4.1) and we investigate the so called macro program problem: Does there exist an algorithm which decides for any given program whether it is a macro programm or not?

In order to shorten our proofs we restrict ourselves to the language ALGOL 60-P introduced in [4]. ALGOL 60-P has the following main restrictions and modifications compared to ALGOL 60 [5]:

a) Only proper procedures, no function procedures are allowed.

b) Specification and value parts in procedure headings are empty.

c) Only identifiers are allowed as actual parameters of procedure statements.

d) Beside **begin** and **end** we have an additional pair of *statement braces* { }. They act as block-**begin** and block-**end**. We require that all procedure bodies are included in these braces and we call them *body braces* in this context. In other contexts they are called *call braces*.

e) The only data types are **real** and **bool**. Declarators in type declarations are written **ref real** and **ref bool** instead of **real** and **Boolean**.

f) We exclude arrays, subscripted variables, switches, switch designators, and unsigned integers as labels.

See [4] for a more complete definition of ALGOL 60-P.

In [4] the notion of a *syntactical* ALGOL 60-P program is introduced. A syntactical program Π is called *formal* if there is for every occurrence (i, Z_i) of an identifier Z_i in Π exactly one defining occurrence $(j, Z_j) = \delta(i, Z_i)$, $Z_j = Z_i$, in Π. i and j represent the occurrence positions of the identifiers Z_i and Z_j in Π. Two formal programs are called *identical* if they differ only by an *admissible* renaming of identifiers. A formal program is called *distinguished* if different defining occurrences of identifiers $(i, Z_i) \neq (j, Z_j)$ are denoted by different identifiers $Z_i \neq Z_j$. A formal program is called *compilable* if any applied occurrence (i, Z_i) of an identifier, bound by the defining occurrence $\delta(i, Z_i) = (j, Z_j)$, is applied appropriately according to the definition. The notion of a *partially compilable* program may also be introduced and we may say when a program Π' *results from another program Π by application of the copy rule* $(\Pi \vdash \Pi')$. If $\Pi \overset{+}{\vdash} \Pi'$ then Π is partially compilable and Π' is at least formal. If a formal program Π is *original*, i.e. a program without call braces, then the set

$$E_\Pi := \{\Pi' \mid \Pi \overset{*}{\vdash} \Pi'\}$$

is called the *execution* of Π. E_Π contains the *execution tree* T_Π, and it is defined when an original program is called to have *formally correct parameter transmissions*. By the help of the *reduced execution* $E_{r\Pi}$ or the *reduced execution tree* $T_{r\Pi} \subseteq E_{r\Pi}$ we may say when two original programs are *formally equivalent*. The program properties to have formally correct parameter transmissions and to be a macro program are invariant for formally equivalent programs (see Theorem 4 in [4]).

In 2. we prove theorems on elimination of global procedure parameters in ALGOL 60-P programs. There exists an effective process which constructs for every original program without global formal procedure parameters a formally equivalent one without any procedure nesting (Theorem 2.4). The same is true for original programs without procedure identifiers as actual parameters (Theorem 2.6). We do not have such an effective process for the class of all ALGOL 60-P programs (see Theorem 7 in [4]). Beyond this, in Theorem 2.7 we present an example of an original program which is not formally equivalent to any program without global formal procedure parameters. So the "expressional power" of ALGOL 60-P diminishes as soon as procedure nestings are disallowed.

In 3. we generalize ALGOL 60-P to ALGOL 60-P-G such that ALGOL 60-P is a sublanguage. In ALGOL 60-P-G additional formal and actual procedure parameters of new kind are introduced. All notions for ALGOL 60-P may be readily transferred to ALGOL 60-P-G. There is an effective process which constructs for every ALGOL 60-P-G program a formally equivalent one without any procedure nesting (Theorem 3.3). So the expressional power of ALGOL 60-P-G does not diminish when procedure nestings are disallowed. We say ALGOL 60-P-G has the modularity property whereas ALGOL 60-P does not have this property (Theorem 3.4). In proofs on executions of programs procedure nestings are highly disturbing as applications of the copy rule yield additional procedure declarations. The importance of ALGOL 60-P-G lies in the fact that we may restrict ourselves to programs without procedure nestings.

In 4. we show that the macro program problem is algorithmically unsolvable for ALGOL 60-P-G (Theorem 4.3). The unsolvability is reduced to the unsolvability of the halting problem for a special type of two tape pushdown automata (Theorem 4.2). The Theorems in 2. and 3. allow to show that the macro program problem is also algorithmically unsolvable for ALGOL 60-P, even if we restrict ourselves to compilable programs which have formally correct parameter transmissions (Theorem 4.4). The unsolvability holds also for ALGOL 60, PL/1, and ALGOL 68. Two premisses are responsible for the unsolvability of the macro program problem in ALGOL 60-P: 1. ALGOL 60-P allows procedure nestings and 2. allows "naked" procedure identifiers as actual parameters. We have solvability if one of these premisses is invalid (Theorem 4.1 and its Corollary).

In 5. macro grammars theory is applied to the macro program problem. Macro grammars are highly related to ALGOL programs with procedures. Several solvability results on macro grammars are known in the literature (Theorem 5.1, Aho [1], Fischer [3], Rounds [7]). Therefore the unsolvability results in Theorem 4.3 and 4.4 are surprising. But, actually, macro grammars represent only special ALGOL 60-P-G programs, namly without procedure nesting and without procedure parameters of old kind. The macro program problem is solvable for this class of special programs (Theorem 5.2). Theorem 5.3 and 5.4 show that the premisses for the decidability in Theorem 5.2 and 4.1 may be essentially weakened. Especially Theorem 5.4 whose proof combines Fischer's and our methods can be applied to show a conjecture presented in [5]: The macro program problem is solvable for ALGOL programs without global formal procedure parameters, but under inclusion of procedure nestings and of expressions and statements as actual parameters (Theorem 5.5).

At the end of this paper we have listed some open questions.

2. Elimination of Global Procedure Parameters in ALGOL 60-P Programs

We deal with programs in ALGOL 60-P

At first we consider an original, distinguished program Π. Let (i, f) and (j, g) be two defining occurrences of non-formal identifiers f and g, such that the scope of f is contained in the scope of g.[1] Then we define a one-element relation in Π

$$R_\Pi := \{((i, f), (j, g))\}.$$

Definition 2.1. We consider T_Π. We *extend the relation* R_Π *to all programs* $\Pi' \in T_\Pi$ by induction. Let Π'' be a generated program in T_Π, let $R_{\Pi'}$ be defined already for the immediate predecessor $\Pi' \vdash \Pi''$. We may transfer the relation $R_{\Pi'}$

[1] The *scope* of an identifier f is the smallest block containing the defining occurrence δf of this identifier.

in a natural way from Π' to Π'' and we denote this transferred relation in Π'' also by $R_{\Pi'}$.

a) If for $\Pi' \vdash \Pi''$ in Π' a copy σ' of a procedure σ is called which does not properly contain the declaration φ of f, then we define

$$R_{\Pi''} := R_{\Pi'}.$$

b) If a copy σ' of a procedure σ is called which properly contains the declaration φ of f and which does not properly contain the declaration γ of g and if a relation $(i', f') R_{\Pi'} (j', g')$ holds where (i', f') is the defining occurrence of the identifier f' of that copy φ' of φ contained in σ', then we define

$$R_{\Pi''} := R_{\Pi'} \cup \{((i'', f''), (j', g'))\}.$$

(i'', f'') is a defining occurrence and is the modification of (i', f') in the modified body of σ', which replaces the procedure statement calling σ'.

c) If a copy σ' of a procedure σ is called which properly contains the declaration γ of g (and consequently properly contains the declaration φ of f) and if a relation $(i', f') R_{\Pi'} (j', g')$ holds where (i', f') and (j', g') are the defining occurrences of the identifiers of those copies φ' and γ' of φ and γ contained in σ', then we define

$$R_{\Pi''} := R_{\Pi'} \cup \{((i'', f''), (j'', g''))\}.$$

(i'', f'') and (j'', g'') are defining occurrences and are the modifications of (i', f') and (j', g') in the modified body of σ', which replaces the procedure statement calling σ'.

Lemma 2.1. a) For all $\Pi' \in T_\Pi$ $R_{\Pi'}$ is a function defined on the set of all defining occurrences of identifiers of copies of φ and mapping onto the set of all defining occurrences of identifiers of copies of γ.

b) R'_Π may be written as a disjoint union ${}^1R'_\Pi \cup {}^2R'_\Pi$ where ${}^1R'_\Pi$ is a 1-1 mapping onto the set of those definining occurrences of identifiers of copies of γ which are local to some procedure body.

c) If $(i', f') R_{\Pi'} (j', g')$ holds then the scope of f' is contained in the scope of g'. If the scope of f is properly contained in the scope of g then all containments of scopes are proper.

Let Π be a distinguished original program of the form

$$\textbf{begin} \ldots \underbrace{\textbf{proc}\, \bar{f}\, (\bar{x}_1, \ldots, \bar{x}_n); \{\bar{\varrho}\}}_{\bar{\varphi}}; \ldots;$$

$$\underbrace{\textbf{proc}\, g\, (y_1, \ldots, y_m); \{\ldots \underbrace{\textbf{proc}\, f(x_1, \ldots, x_n); \{\varrho\}}_{\varphi}; \ldots\}}_{\psi}; \ldots \textbf{end}$$

The procedures $\bar{\varphi}$ and ψ are declared within the same block; the smallest surrounding procedure of φ is ψ. $\bar{\varphi}$ is generated from φ by an admissible renaming of identifiers in φ which are local to φ, f included. (This means especially that $y_1, \ldots, y_m$ do not occur in φ, f in φ becomes $\bar{f}$, $\bar{f}$ in φ remains $\bar{f}$, all identifiers global to φ remain unchanged and their defining occurrences are outside the extended body of ψ.)

Theorem 2.1. Let (i, f) denote an applied occurrence of f in ψ and let the distinguished, original program $\tilde{\Pi}$ be generated from Π by a renaming of some (i, f)-s to $(i, \bar{f})$-s. Then Π and $\tilde{\Pi}$ are formally equivalent.

In order to prove this Theorem 2.1 we consider the one-element relation

$$R_\Pi := \{(\delta f, \delta \bar{f})\}\ ^2$$

in Π and we extend this relation to all programs $\Pi' \in T_\Pi$ due to Definition 2.1.

Lemma 2.2. If $(i', f')\, R_\Pi\, (\bar{i}', \bar{f}')$ holds then the copy $\bar{\varphi}'$ of $\bar{\varphi}$ denoted by $(\bar{i}', \bar{f}')$ can be generated from the copy φ' of φ denoted by (i', f') by an admissible renaming of identifiers in φ' which are local to φ', f' included.

For $\tilde{\Pi}$ we have analogous relations $R_{\tilde{\Pi}'}$, $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ and an analogous Lemma 2.2.

Lemma 2.3. We consider T_Π and $T_{\tilde{\Pi}}$. Then for every program $\Pi' \in T_\Pi$ there exists exactly one ("nearly identical") program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ with the following properties:

1) Π' and $\tilde{\Pi}'$ are identical with the exception of some renamings of applied occurrences (j', f') of identifiers of copies φ' of φ in Π'.

2) (j', f') is renamed to an applied occurrence $(j', \bar{f}')$ of the identifier of a copy $\bar{\bar{\varphi}}'$ of $\bar{\bar{\varphi}}$ in $\tilde{\Pi}'$. $\bar{\bar{\varphi}}$ is that procedure which stands in $\tilde{\Pi}'$ on the "same place" where $\bar{\varphi}$ in Π' stands.

3) In Π' the relation

$$\delta(j', f')\, R_{\Pi'}\, \delta(j', \bar{f}')$$

holds. $\delta(j', \bar{f}')$ is primarily a defining occurrence in $\tilde{\Pi}'$, but because of property 1) also a defining occurrence in Π'.

4) The relations $R_{\Pi'}$ and $R_{\tilde{\Pi}'}$ are identical.

Vice versa, for every program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ there exists exactly one program $\Pi \in T_\Pi$ with the same properties 1)–4).

Proof. For $\Pi' \in T_\Pi$ we have at most one $\tilde{\Pi}' \in T_{\tilde{\Pi}}$. If there would be another $\tilde{\Pi}_1' \in T_{\tilde{\Pi}}$ then $\tilde{\Pi}'$ and $\tilde{\Pi}_1'$ would have the same reduced main program [4] such that $\tilde{\Pi}'$ and $\tilde{\Pi}_1'$ would be identical.

We consider $\Pi \in T_\Pi$; then $\tilde{\Pi} \in T_{\tilde{\Pi}}$ is a program with the desired properties 1)–4).

Let now $\Pi'' \in T_\Pi$ be a generated program with the immediate predecessor $\Pi' \vdash \Pi''$. Let Π'' result from Π' by the procedure statement

$$h(a_1, \ldots, a_\nu)$$

2 For our convenience we have dropped i in (i, f) and $(i, \bar{f})$.

in Π'. By induction hypothesis there exists a nearly identical program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$. Since Π' is partially compilable $\tilde{\Pi}'$ is partially compilable, too, because of property 1) and 2). In $\tilde{\Pi}'$ we have on the same spot as $h(a_1, \ldots, a_\nu)$ in Π' a procedure statement

$$\tilde{h}(\tilde{a}_1, \ldots, \tilde{a}_\nu)$$

with the same number ν of parameters, and this statement generates $\tilde{\Pi}'' \dashv \tilde{\Pi}'$.

Indifferently, whether h is properly renamed to $\tilde{h}$ or not we may infer from Lemma 2.2 and property 1)–4) for Π' and $\tilde{\Pi}'$ to property 4) for Π'' and $\tilde{\Pi}''$.

Let now h be properly renamed to $\tilde{h}$. Then $h = f'$ denotes a copy φ' of φ in Π' and $\tilde{h} = \bar{f}'$ fenotes a copy $\tilde{\bar{\varphi}}'$ of $\tilde{\bar{\varphi}}$ in Π' with $\delta f' R_{\Pi'} \delta \bar{f}'$ in Π'. We consider the modified bodies of φ' and $\tilde{\bar{\varphi}}'$, which replace $h(a_1, \ldots, a_\nu)$ and $\tilde{h}(\tilde{a}_1, \ldots, \tilde{a}_\nu)$ in Π'' and $\tilde{\Pi}''$. Proper renamings may come into these modified bodies firstly by properly renamed actual parameters a_ι, $\tilde{a}_\iota$, secondly by different non-formal identifiers g, $\tilde{g}$ in the bodies of φ' and $\tilde{\bar{\varphi}}'$. These are global to the bodies because of Lemma 2.2 and property 1) and 2) for Π' and $\tilde{\Pi}'$. Both for Π'' and $\tilde{\Pi}''$ situation a) in the definition of $R_{\Pi''}$ and $R_{\tilde{\Pi}''}$ applies.

In the first case the induction hypothesis $\delta a_\iota R_{\Pi'} \delta \tilde{a}_\iota$ in Π' leads to $\delta a_\iota R_{\Pi''} \delta a_\iota$ in Π'' by definition of $R_{\Pi''}$. In the second case we have a $\bar{g}$ in Π' on the same spot as $\tilde{g}$ in $\tilde{\Pi}'$. If $\bar{g} = g$ then by induction hypothesis $\delta \bar{g} R_{\Pi'} \delta \tilde{g}$ holds in Π' such that $\delta g R_{\Pi''} \delta \tilde{g}$ holds in Π'' by definition of $R_{\Pi''}$. If $\bar{g}$ is different from g then by Lemma 2.2 $\delta g R_{\Pi'} \delta \bar{g}$ with $g = f'$ and $\bar{g} = \bar{f}'$ holds in Π'. Then $\bar{g} = \tilde{g}$ holds such that $\delta g R_{\Pi''} \delta \tilde{g}$ is true by definition of $R_{\Pi''}$. Otherwise $\delta \bar{g} R_{\Pi'} \delta \tilde{g}$ would hold by induction hypothesis such that $\bar{g}$ would be the identifier of a copy both of φ and $\bar{\varphi}$ which is wrong.

Let now h and $\tilde{h}$ be identical. h denotes a procedure σ' in Π' and $\tilde{h}$ a nearly identical procedure $\tilde{\sigma}'$ in $\tilde{\Pi}'$ on the same spot as σ' in Π'. We consider the modified bodies of σ' and $\tilde{\sigma}'$, modified by the copy rule. Proper renamings may come into these modified bodies firstly by properly renamed actual parameters a_ι, $\tilde{a}_\iota$, secondly by different non-formal identifiers g, $\tilde{g}$ which are global or local to the bodies of σ' and $\tilde{\sigma}'$.

In the first case we argue as before. In the second case we have $\delta g R_{\Pi'} \delta \tilde{g}$ in Π' by induction hypothesis. If g is global then by Lemma 2.1 b) $\tilde{g}$ is global too and we have $\delta g R_{\Pi''} \delta \tilde{g}$ by definition of $R_{\Pi''}$. But, even if g is local, then the definition of $R_{\Pi''}$ guarantees that the properties 1)–3) hold for Π'' and $\tilde{\Pi}''$, too.

The proof for the "vice versa" direction is similar. Q.e.d.

Theorem 2.1 is an immediate consequence of Lemma 2.3.

Theorem 2.2. Let Π be a distinguished, original program of the form

$$\textbf{begin} \ldots \underbrace{\textbf{proc}\, g\, (y_1, \ldots, y_m); \{\ldots \underbrace{\textbf{proc}\, f\, (x_1, \ldots, x_n); \{\varrho\}}_{\varphi}; \ldots\}}_{\psi}; \ldots \textbf{end}$$

Let all global parameters $\neq f$ of φ be global to the extended body[3] of ψ. If we remove φ and put it immediately in front of ψ, then we get a program $\tilde{\Pi}$ which is formally equivalent to Π.

Proof. Let Π be given. We create a procedure $\bar{\varphi}$ from φ as in Theorem 2.1 and put it immediately in front of ψ. This new program $\hat{\Pi}$ is obviously formally equivalent to Π because no copy of $\bar{\varphi}$ is called in $T_{\hat{\Pi}}$. Now, we rename all applied occurrences of f to $\bar{f}$ in the main part of ψ. By Theorem 2.1 we get a formally equivalent program $\tilde{\tilde{\Pi}}$ where no copy of φ is called in $T_{\tilde{\tilde{\Pi}}}$. So we may remove φ from $\tilde{\tilde{\Pi}}$ and we get a program which is identical with $\tilde{\Pi}$ and formally equivalent to Π. Q.e.d.

Let $\hat{\Pi}$ be a distinguished original program of the form

$$\textbf{begin} \ldots \underbrace{\textbf{proc}\; x_0(x_1, \ldots, x_n); \{\ldots a \ldots\}}_{\hat{\varphi}}; \ldots \textbf{end}$$

We assume that in the body of $\hat{\varphi}$ there is at least one applied occurrence of the non-formal identifier a which is global to the body of $\hat{\varphi}$ (a might be x_0), i.e. a is a global parameter of $\hat{\varphi}$.

Immediately behind the very first **begin** of $\hat{\Pi}$ we put a defining occurrence of a non-formal identifier $\hat{a}$ with any declaration, e. g.

$$\textbf{ref real} \text{ or } \textbf{proc}\; \hat{a}; \{\,\}.$$

For every procedure declaration

$$\textbf{proc}\; y_0\,(y_1, \ldots, y_m); \{\ldots\}$$

we add $m+1$ new *accompanying formal parameters*

$$\textbf{proc}\; y_0\,(\bar{y}_0, y_1, \bar{y}_1, \ldots, y_m, \bar{y}_m); \{\ldots\} \qquad (*)$$

and for every procedure statement

$$u_0\,(u_1, \ldots, u_m)$$

we add $m+1$ *accompanying actual parameters*

$$u_0\,(\bar{\bar{u}}_0, u_1, \bar{\bar{u}}_1, \ldots, u_m, \bar{\bar{u}}_m)$$

of the following form:

a) If u_ι is equal x_0 then $\bar{\bar{u}}_\iota$, is equal a.

b) If u_ι is non-formal but unequal x_0 then $\bar{\bar{u}}_\iota$ is equal $\hat{a}$.

c) If u_ι is formal and consequently equal a $y_\varkappa$ of $(*)$ then $\bar{\bar{u}}_\iota$ is equal $\bar{y}_\varkappa$.

The created program Π is obviously formally equivalent to $\hat{\Pi}$. In $\Pi' \in T_\Pi$ there are no applied occurrences of the first accompanying formal parameters $\bar{y}_0'$ of the procedure identifiers y_0'.

3 The *extended body* of a procedure is the body extended by the parameter part, **proc**-symbol and procedure identifier excluded. Extended bodies act as blocks (see [4]). An applied occurring of an identifier in the body of a procedure φ whose defining occurrence lies outside the extended body of φ is called a *global parameter* of φ. Thus, the identifier f denoting φ is a global parameter of φ, if f occurs in the body of φ.

Theorem 2.3. Let us rename certain applied occurrences of a to x_0 in the body of that procedure φ denoted by x_0. We get a new program $\tilde{\Pi}$. Then the distinguished original programs Π and $\tilde{\Pi}$ are formally equivalent.

With other words: Global non-formal parameters of a procedure may be replaced by formal parameters. With the help of Theorem 2.2 we may conclude

Theorem 2.4. There is an effective process which constructs for every original program without global formal procedure parameters[4] a formally equivalent one without any procedure nesting.

One remark concerning Theorem 2.3 will be useful in 5.: If procedure $\hat{\varphi}$ in $\hat{\Pi}$ denoted by x_0 has at least one formal parameter then no parameterless procedure y_0 needs any accompanying formal parameter $\bar{y}_0$ and no procedure statement u_0 without actual parameters needs any accompanying actual parameter $\bar{\bar{u}}_0$.

Let α and $\tilde{\alpha}$ be the declarations of a in Π and $\tilde{\Pi}$. In order to prove Theorem 2.3 we consider the one-element relations

$$R_\Pi := \{(\delta x_0, \delta a)\}, \qquad R_{\tilde{\Pi}} := \{(\delta x_0, \delta a)\}$$

in Π and $\tilde{\Pi}$ and we extend these relations to all programs $\Pi' \in T_\Pi$ and $\tilde{\Pi}' \in T_{\tilde{\Pi}}$.

Lemma 2.4. For any procedure statement

$$u_0(\check{u}_0, u_1, \check{u}_1, \ldots, u_\mu, \check{u}_\mu)$$

in $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ the following conditions hold:

I) If u_ι is the identifier x_0' of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$ then $\delta u_\iota R_{\tilde{\Pi}'} \delta \check{u}_\iota$ or $\check{u}_\iota$ is the first accompanying formal parameter $\bar{x}_0'$ of u_ι ($u_\iota F \check{u}_\iota$ for short). $\tilde{\varphi}$ is the procedure declaration of x_0 in $\tilde{\Pi}$.

II) If u_ι is non-formal but does not denote any copy of $\tilde{\varphi}$ then $\check{u}_\iota = \hat{a}$.

III) If u_ι is the identifier $\bar{x}_0'$ of a copy $\tilde{\xi}'$ of $\tilde{\xi}$ then if $a = x_0 > (\check{u}_\iota = u_\iota$ or $\check{u}_\iota F u_\iota)$ and if $a \neq x_0 > \check{u}_\iota = \hat{a}$. $\tilde{\xi}$ is the declaration of the formal parameter $\bar{x}_0$ in $\tilde{\Pi}$.

IV) If u_ι is formal but does not denote any copy of $\tilde{\xi}$ then u_ι is a formal parameter $y_\varkappa'$ and $\check{u}_\iota$ is the accompanying formal parameter $\bar{y}_\varkappa'$.

Proof. Lemma 2.4 is evident for $\tilde{\Pi}$. Let now $\tilde{\Pi}''$ be a generated program in $T_{\tilde{\Pi}}$ and we assume that Lemma 2.4 is already proved for the immediate predecessor $\tilde{\Pi}' \vdash \tilde{\Pi}''$. Let $\tilde{\Pi}''$ result from $\tilde{\Pi}'$ by the procedure statement

$$u_0\ (\check{u}_0, u_1, \check{u}_1, \ldots, u_m, \check{u}_m)$$

in $\tilde{\Pi}'$. If there is a procedure statement

$$u_0''\ (\check{u}_0'', u_1'', \check{u}_1'', \ldots, u_\mu'', \check{u}_\mu'')$$

in the new modified procedure body in $\tilde{\Pi}''$ then there is a corresponding procedure statement

$$u_0'\ (\check{u}_0', u_1', \check{u}_1', \ldots, u_\mu', \check{u}_\mu')$$

in the body of the procedure denoted by u_0.

4 A global parameter of a procedure φ which is a formal parameter of a larger procedure ψ is called a *global formal parameter* of φ.

We discuss the three different situations a), b), c) in the definition for $R_{\tilde{\Pi}''}$.

a) I) Let u'_ι be the identifier x'_0 of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$. Then u'_ι is global to the body of the procedure denoted by u_0. Then $u''_\iota = u'_\iota$.

(1) If $\delta u'_\iota R_{\tilde{\Pi}'} \delta \breve{u}'_\iota$ then by Lemma 2.1 $\breve{u}'_\iota$ is also global to the body of the procedure denoted by u_0. Then $\breve{u}''_\iota = u'_\iota$. Then $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u'_\iota F \breve{u}'_\iota$ then $u_0 = u'_\iota$. Then $\breve{u}'_\iota$ is replaced by $\breve{u}_0$, $\breve{u}''_\iota = \breve{u}_0$ and $\breve{u}''_\iota$ is non-formal. Then $\delta u_0 R_{\tilde{\Pi}'} \delta \breve{u}_0$ and $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$ by definition of $R_{\tilde{\Pi}''}$.

II) Let u'_ι be non-formal but not denote any copy of $\tilde{\varphi}$. Then $\breve{u}'_\iota = \hat{a}$. Then u''_ι does not denote any copy of $\tilde{\varphi}$ and $\breve{u}''_\iota = \hat{a}$.

III) Let u'_ι be the identifier $\bar{x}'_0$ of a copy $\tilde{\xi}'$ of $\tilde{\xi}$. Then $u_0 F u'_\iota$ and u_0 is the identifier x'_0 of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$. Then $\delta u_0 R_{\tilde{\Pi}'} \delta \breve{u}_0$. u'_ι is replaced by $\breve{u}_0$, $u''_\iota = \breve{u}_0$.

(1) If $a = x_0$ then $R_{\tilde{\Pi}'}$ is the identical function and $u_0 = \breve{u}_0$.

(1.1) If $\breve{u}'_\iota = u'_\iota$ then $\breve{u}''_\iota = u''_\iota = \breve{u}_0$. Then $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$.

(1.2) If $\breve{u}'_\iota F u'_\iota$ then $u_0 = \breve{u}'_\iota$ and $\breve{u}''_\iota = \breve{u}'_\iota$. Then $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$.

(2) If $a \neq x_0$ then $\breve{u}'_\iota = \hat{a}$. u''_ι does not denote any copy of $\tilde{\varphi}$ and $\breve{u}''_\iota = \hat{a}$.

(IV) Let u'_ι be formal but not denote any copy of $\tilde{\xi}$. Then u'_ι is equal a $y'_\varkappa$ and $\breve{u}'_\iota$ is equal $\bar{y}'_\varkappa$.

(1) If u'_ι is local to the body of u_0 then so is $\breve{u}'_\iota$. Then u''_ι and $\breve{u}''_\iota$ are formal and local to the modified body of u_0, u''_ι is equal a $y''_\varkappa$ and $\breve{u}''_\iota$ is equal $\bar{y}''_\varkappa$. u''_ι does not denote any copy of $\tilde{\xi}$.

(2) If u'_ι is a formal parameter of u_0 then so is $\breve{u}'_\iota$. We have $u''_\iota = u_\varkappa$ and $\breve{u}''_\iota = \breve{u}_\varkappa$ with $\varkappa \geqq 1$. $u_\varkappa$ and $\breve{u}_\varkappa$ are non-formal.

(2.1) If u'' is the identifier x'_0 of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$ then $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$ by definition of $R_{\tilde{\Pi}''}$.

(2.2) If u''_ι does not denote any copy of $\tilde{\varphi}$ then $\breve{u}''_\iota = \hat{a}$.

b) I) Let u'_ι be the identifier x'_0 of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$.

(1) If $\delta u'_\iota R_{\tilde{\Pi}'} \delta \breve{u}'_\iota$ then $\breve{u}'_\iota$ is non-formal and global to the body of u_0. Then is $\breve{u}''_\iota = \breve{u}'_\iota$.

(1.1) If u'_ι is global, too, then is $u''_\iota = u'_\iota$. Then $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}'_\iota$ by definition of $R_{\tilde{\Pi}''}$.

(1.2) If u'_ι is local, then u''_ι is the identifier x''_0 of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ with $\delta u''_\iota R_{\tilde{\Pi}''} \delta \breve{u}''_\iota$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u'_\iota F \breve{u}'_\iota$ then $\breve{u}'_\iota$ is formal. Then both u'_ι and $\breve{u}'_\iota$ are local to the body of u_0. Then u''_ι is the identifier x''_0 of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ and $u''_\iota F \breve{u}''_\iota$.

II) Let u'_ι be non-formal but not denote any copy of $\tilde{\varphi}$. Here we conclude as in case a) II).

III) Let u'_ι be the identifier $\bar{x}'_0$ of a copy $\tilde{\xi}'$ of $\tilde{\xi}$. u'_ι is local to the body of u_0. u''_ι is the identifier $\bar{x}''_0$ of a copy $\tilde{\xi}''$ of $\tilde{\xi}$.

(1) If $a = x_0$ and

(1.1) if $\breve{u}'_\iota = u'_\iota$, then $\breve{u}''_\iota = u''_\iota$.

(1.2) If $\breve{u}'_\iota F u'_\iota$, then $\breve{u}'_\iota$ is local to the body of u_0. Then $\breve{u}''_\iota$ is the identifier x''_0 of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ and $\breve{u}''_\iota F u''_\iota$.

(2) If $a \neq x_0$ then $\check{u}' = \hat{a}$. Then $\check{u}_\iota'' = \check{u}_\iota' = \hat{a}$.

IV) Let u_ι' be formal but not denote any copy of $\tilde{\xi}$. Here we conclude as in case a) IV).

c) I) Let u_ι' be the identifier x_0' of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$.

(1) If $\delta u_\iota' R_{\tilde{\Pi}'} \delta \check{u}_\iota'$ and

(1.1) if u_ι' is global to the body of u_0, then so is $\check{u}_\iota'$. Then $\check{u}_\iota'' = \check{u}_\iota'$ and $u_\iota'' = u_\iota'$. Then $\delta u_\iota'' R_{\tilde{\Pi}''} \delta \check{u}_\iota''$.

(1.2) If u_ι' is local then so is $\check{u}_\iota'$. Then u_ι'' is the identifier x_0'' of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ with $\delta u_\iota'' R_{\tilde{\Pi}''} \delta \check{u}_\iota''$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u_\iota' F \check{u}_\iota'$ then both u_ι' and $\check{u}_\iota'$ are local to the body of u_0. Then u_ι'' is the identifier x_0'' of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ with $u_\iota'' F \check{u}_\iota''$.

II) Let u_ι' be non-formal and not denote any copy of $\tilde{\varphi}$. Here we conclude as in case a) II).

III) Let u_ι' be the identifier $\tilde{x}_0'$ of a copy $\tilde{\xi}'$ of $\tilde{\xi}$. Here we conclude as in case b) III).

IV) Let u_ι' be formal but not denote any copy of $\tilde{\xi}$. Here we conclude as in case a) IV). Q.e.d.

Lemma 2.5. We consider T_Π and $T_{\tilde{\Pi}}$. Then for every program $\Pi' \in T_\Pi$ there exists exactly one ("nearly identical") program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ with the following properties:

1) Π' and $\tilde{\Pi}'$ are identical with the exception of some renamings of applied occurrences (j', a') of identifiers of copies α' of α in Π'.

2) (j', a') is renamed to an applied occurrence $(j', \tilde{x}_0')$ of the identifier of a copy $\tilde{\xi}'$ of $\tilde{\xi}$ in $\tilde{\Pi}'$.

3) In Π' the relation

$$\pi \delta(j', \tilde{x}_0') R_{\Pi'} \delta(j', a')$$

holds. $\delta(j', \tilde{x}_0')$ is primarily a defining occurrence in $\tilde{\Pi}'$ but because of 1) also a defining occurrence in Π'. π is an operator which yields for any defining occurrence of a formal parameter the defining occurrence of the identifier of the associated procedure.

4) The relations $R_{\Pi'}$ and $R_{\tilde{\Pi}'}$ are identical.

Vice versa, for every program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ there exists exactly one program $\Pi' \in T_\Pi$ with the same properties 1)–4).

Proof. For $\Pi' \in T_\Pi$ we have at most one $\tilde{\Pi}' \in T_{\tilde{\Pi}}$. We consider $\Pi \in T_\Pi$; then $\tilde{\Pi} \in T_{\tilde{\Pi}}$ is a program with the desired properties.

Let now $\Pi'' \in T_\Pi$ be a generated program with the immediate predecessor $\Pi' \vdash \Pi''$. Let Π'' result from Π' by the procedure statement

$$u_0(\check{u}_0, u_1, \check{u}_1, \ldots, u_m, \check{u}_m)$$

in Π'. By induction hypothesis there exits a nearly identical program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$. Since Π' is partially compilable $\tilde{\Pi}'$ is partially compilable, too, because of 1) and 2). In $\tilde{\Pi}'$ we have

$$u_0(\check{u}_0, u_1, \check{u}_1, \ldots, u_m, \check{u}_m)$$

on the same spot as $u_0(\check{u}_0, \ldots, \check{u}_m)$ in Π' and this statement generates $\tilde{\Pi}'' \dashv \tilde{\Pi}'$. At first we can infer from 1)–4) for Π' and $\tilde{\Pi}'$ to 4) for Π'' and $\tilde{\Pi}''$.

We must now look at renamings in the bodies of those procedures in Π' and $\tilde{\Pi}'$ which are denoted by u_0. If we have such a renaming (j', a') to $(j', \bar{x}_0')$ with $\pi\delta\bar{x}_0' R_{\Pi'}\delta a'$ we discuss the three different situations a), b), c) in the definitions for $R_{\Pi''}$ and $R_{\tilde{\Pi}''}$.

a) u_0 is the identifier x_0' of copies φ' and $\tilde{\varphi}'$ of φ and $\tilde{\varphi}$ with $\delta x_0' = \pi\delta\bar{x}_0'$. a' is global to the body of $u_0 = x_0'$ and does not change, $\bar{x}_0'$ is replaced by $\check{u}_0$. By Lemma 2.4 $\delta u_0 R_{\Pi'} \delta\check{u}_0$ holds. By Lemma 2.1, a) we have $\check{u}_0 = a'$. The renaming is cancelled.

b) $\pi\delta\bar{x}_0' = \delta x_0'$ is local to the body of u_0. a' is global to the body of u_0 and does not change. $\bar{x}_0'$ is modified to $\bar{x}_0''$ with $\pi\delta\bar{x}_0'' R_{\Pi''}\delta a'$ by definition of $R_{\Pi''}$.

c) $\pi\delta\bar{x}_0' = \delta x_0'$ and a' are local to the body of u_0. $\bar{x}_0'$ and a' are modified to $\bar{x}_0''$ and a'' with $\pi\delta x_0'' R_{\Pi''}\delta a''$ by definition of $R_{\Pi''}$.

The "vice versa" direction is proved similarly. Q.e.d.

Theorem 2.3 is an immediate consequence of Lemma 2.5.

Now we should like to prove Theorem 2.3 if a is formal. But the additional assumption is necessary that x_0 is never used as an actual parameter. For this purpose we drop the accompanying formal parameters $\bar{y}_1, \ldots, \bar{y}_m$ and the accompanying actual parameters $\bar{u}_1, \ldots, \bar{u}_m$. $\bar{u}_0$ has the following form:

a) If u_0 is equal x_0 then $\bar{u}_0$ is equal a.

b) If u_0 is unequal x_0 then $\bar{u}_0$ is equal $\hat{a}$.

The created program Π is formally equivalent to $\hat{\Pi}$. In $\Pi' \in T_\Pi$ there are no applied occurrences of accompanying formal parameters $\bar{y}_0'$.

Theorem 2.5. Let a be formal and x_0 never be used as an actual parameter in Π. Let us rename certain occurrences of a to $\bar{x}_0$ in the body of that procedure denoted by x_0. We get a new program $\tilde{\Pi}$. Then the distinguished original programs $\hat{\Pi}$, Π, and $\tilde{\Pi}$ are formally equivalent.

With the help of Theorem 2.3, a generalized Theorem 2.5 which allows to eliminate a global procedure parameter which occurs simultanously in several "parallel" procedures, and with the help of a generalized Theorem 2.2 which allows to move several "parallel" procedures simultanously we may conclude

Theorem 2.6. There in an effective process which constructs for every original program in which procedure identifiers do not occur as actual parameters a formally equivalent program without any procedure nesting.

We consider a distinguished original program Π. Let (i, f) and (j, g) be two defining occurrences of a non-formal identifier f and a formal identifier g, such that the scope of f is contained in the scope of g. We define

$$R_\Pi := \{((i, f), (j, g))\}.$$

Definition 2.2. The *extension* of R_Π is defined by induction:

a) If for $\Pi' \vdash \Pi''$ a copy σ' of a procedure σ is called which does not properly contain the declaration φ of f, then we define

$$R_{\Pi''} := R_{\Pi'}.$$

b) If a copy σ' of a procedure σ is called which properly contains the declaration φ of f and which is properly contained in the scope of g and if a relation $(i', f') R_{\Pi'} (j', u')$ holds where (i', f') is the defining occurrence of the identifier f' of that copy φ' of φ contained in σ', then we define

$$R_{\Pi''} := R_{\Pi'} \cup \{((i'', f''), (j', u'))\}.$$

(i'', f'') is a defining occurrence and is the modification of (i', f') in the modified body of σ', which replaces the procedure statement calling σ'.

c) If by $s(u_1, \ldots, u_\nu)$ a copy σ' of the procedure σ is called whose formal parameter is g (and which consequently properly contains the declaration φ of f) and if a relation $(i', f') R_{\Pi'} (j', g')$ holds where (i', f'), (j', g') are the defining occurrences of identifiers of those copies φ' and γ' of φ and γ contained in σ' (γ is the declaration of g), then we define

$$R_{\Pi''} := R_{\Pi''} \cup \{((i'', f''), \delta u_\iota)\}.$$

(i'', f'') is a defining occurrence and is the modification of (i', f') in the modified body of σ', which replaces the procedure statement $s(u_1, \ldots, u_\nu)$. δu_ι is the defining occurrence of the ι-th actual parameter u_ι where g' is the ι-th formal parameter of σ'.

d) If a copy σ' of a procedure σ is called which properly contains the procedure whose formal parameter is g (and consequently properly contains the declaration φ of f) and if a relation $(i', f') R_{\Pi'} (j', g')$ holds where (i', f'), (j', g') are the defining occurrences of the identifiers of those copies φ' and γ' of φ and γ contained in σ', then we define

$$R_{\Pi''} := R_{\Pi'} \cup \{((i'', f''), (j'', g''))\}.$$

(i'', f'') and (j'', g'') are defining occurrences and are the modifications of (i', f') and (j', g') in the modified body of σ' which replaces the procedure statement calling σ'.

Lemma 2.6. a) For every $\Pi' \in T_\Pi$ $R_{\Pi'}$ is a function defined on the set of all defining occurrences of identifiers of copies of φ and mapping into the set of defining occurrences of identifiers.

b) R'_Π may be written as a disjoint union ${}^1R'_\Pi \cup {}^2R'_\Pi$ where ${}^1R'_\Pi$ is a 1-1 mapping onto the set of all defining occurrences of identifiers of copies of γ and where ${}^2R'_\Pi$ is a mapping into the set of defining occurrences of non-formal identifiers global to all procedure bodies.

c) If $(i', f') R_{\Pi'} (j', u')$ holds then the scope of f' is properly contained in the scope of u'.

We return to Theorem 2.5 and consider the one-element relations

$$R_\Pi := \{(\delta x_0, \delta a)\}, \qquad R_{\tilde\Pi} := \{(\delta x_0, \delta a)\}$$

in Π and $\tilde\Pi$. We extend these relations to all programs $\Pi' \in T_\Pi$ and $\tilde\Pi' \in T_{\tilde\Pi}$ by Definition 2.2.

Lemma 2.7. For any procedure statement

$$u_0(\check u_0, u_1, \ldots, u_\mu)$$

in $\tilde\Pi' \in T_{\tilde\Pi}$ the following conditions hold:

I) If u_0 is the identifier x_0' of a copy $\tilde\varphi'$ of $\tilde\varphi$ then we have $\delta u_0 R_{\tilde\Pi'} \delta \check u_0$ or $\check u_0$ is the accompanying formal parameter $\bar x_0'$ of $u_0 (u_0 F \check u_0)$. $\tilde\varphi$ is the procedure declaration of x_0 in $\tilde\Pi$.

II) Otherwise $\check u_0$ is equal $\hat a$.

III) $\check u_0$ and u_ι do not denote any copies of $\tilde\varphi$.

Lemma 2.8. We consider T_Π and $T_{\tilde\Pi}$. Then for every program $\Pi' \in T_\Pi$ there exists exactly one ("nearly identical") program $\tilde\Pi' \in T_\Pi$ with the following properties:

1) Π' and $\tilde\Pi'$ are identical with the exception of at most some renamings of applied occurrences (j', u') of identifiers in Π'.

2) (j', u') is renamed to an applied occurrence $(j', \bar x_0')$ of the identifier of a copy $\tilde\xi'$ of $\tilde\xi$ in $\tilde\Pi'$. $\tilde\xi$ is the declaration of the formal parameter $\bar x_0$ in $\tilde\Pi$.

3) In Π' the relation

$$\pi\delta(j', \bar x_0') R_{\Pi'} \delta(j', u')$$

holds. $\delta(j', \bar x_0')$ is primarily a defining occurrence in $\tilde\Pi'$ but because of 1) also a defining occurrence in Π'.

4) The relations $R_{\Pi'}$ and $R_{\tilde\Pi'}$ are identical.

Vice versa, for every program $\tilde\Pi' \in T_{\tilde\Pi}$ there exists exactly one program $\Pi' \in T_\Pi$ with the same properties 1)–4).

Theorem 2.5 is an immediate consequence of Lemma 2.8. We have omittet proofs of Lemma 2.7 and 2.8 as they are similar to those of Lemma 3.1 and 3.2 of the next paragraph.

In addition to Theorem 2.4 we give the following remarks. In [4], Theorem 7, it is proved that there is no general algorithm which transforms any original program into a formally equivalent one without global formal parameters and consequently one without procedure nesting. We can even prove

Theorem 2.7. There is an original program Π which is not formally equivalent to any original program without global formal procedure parameters.

Proof. We consider the correspondence system $\mathfrak{C}$ with

$$\Gamma = \{\gamma_1 = (c_1, \tilde c_1) = (A, \bar A)\}.$$

$\mathfrak{C}$ has the solutions 1, and 1, 1, and 1, 1, 1 etc. $\Pi^1_{\mathfrak{C}}$ of Theorem 8 in [4] has an infinite execution tree $T_{\Pi^1_{\mathfrak{C}}}$ of the following structure

$$\begin{array}{lll}
\multicolumn{3}{l}{\Pi^1_{\mathfrak{C}} = \ldots L_1(E,\bar{E})\ldots} \\
\multicolumn{3}{l}{\quad\top} \\
\multicolumn{3}{l}{\ldots\{\ldots L_1(A[1],\bar{A}[1]);\; M(A[1],\bar{A}[1])\}\ldots} \\
\top & \qquad\top & \\
\vdots & \ldots\{A[1](\bar{A}[1])\}\ldots & \left.\begin{array}{l}\\ \\ \\ \end{array}\right\} \text{7 nodes, see below} \\
\vdots & \qquad\top & \\
\vdots & \qquad\vdots & \\
\top & \ldots\{\qquad\}\ldots & \\
\multicolumn{2}{l}{\ldots\{\ldots L_1(A[\nu],\bar{A}[\nu]);\; M(A[\nu],\bar{A}[\nu])\}\ldots} & \nu\in\mathbb{N}\setminus\{1\} \\
\top & \qquad\top & \\
\vdots & \ldots\{A[\nu](\bar{A}[\nu])\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{\bar{A}[\nu](A[\nu-1],M,D,D)\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{M(A[\nu-1],\bar{A}[\nu-1])\}\ldots & \\
 & \qquad\top & \\
 & \qquad\vdots & \\
 & \qquad\top & \\
 & \ldots\{M(A[1],\bar{A}[1])\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{A[1](\bar{A}[1])\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{\bar{A}[1](E,M,D,D)\}\ldots & \\
 & \qquad\top & 3\cdot(\nu-1)+5 \text{ nodes} \\
 & \ldots\{M(E,\bar{E})\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{E(\bar{E})\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{\bar{E}(E,D,D,M1)\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{M1(E,\bar{E})\}\ldots & \\
 & \qquad\top & \\
 & \ldots\{\qquad\}\ldots &
\end{array}$$

$T_{\Pi^1_{\mathfrak{C}}}$ has infinitely many nodes which are roots of finite subtrees. The depths of these subtrees are unlimited. E.g. a node $\ldots\{M(A[\nu],\bar{A}[\nu])\}\ldots$, $\nu\in\mathbb{N}$, is root of a linear subtree with $3\cdot\nu+5$ nodes. The *depth* of a tree is the number of nodes in a longest finite $\vdash$-chain if there exists such a chain, otherwise the *depth* is ∞.

If we had a formally equivalent program $\hat{\Pi}$ without global formal parameters then, due to Theorem 2.4, we had a formally equivalent program $\tilde{\Pi}$ without any procedure nesting. We consider nodes which are roots of finite subtrees. The depths of these subtrees are limited by

$$P\cdot G^F+2$$

(see [4]) where P is the number of defining occurrences of non-formal procedure declarations, G is the number of defining occurrences of non-formal identifiers, and F is the number of defining occurrences of formal parameters in $\tilde{\Pi}$. Contradiction! Q.e.d.

3. Generalized ALGOL 60 Programs

We consider the language ALGOL 60-P and we generalize this language to ALGOL 60-P-G such that ALGOL 60-P is a sublanguage.

Procedure headings in ALGOL 60-P-G have the form

$$\mathbf{proc}\ f\ \langle y_1, \ldots, y_{m_f}\rangle\ (x_1, \ldots, x_{n_f}) \qquad (*)$$

$y_1, \ldots, y_{m_f}$ are identifiers. They are called *formal parameters of new kind*. The identifiers $x_1, \ldots, x_{n_f}$ are called *formal parameters of old kind*. When m_f is zero then we drop the brackets $\langle\ \rangle$ as we do with ( ) when n_f is zero. All the other declarations look as in ALGOL 60-P.

Applied occurrences of identifiers in ALGOL 60-P are replaced by so called terms. A *term* is a string generated in a finite process by the following rules:

1. Identifiers are terms.

2. If ψ is an identifier and if $\tau_1, \ldots, \tau_m$, $m \geqq 1$, are terms, then $\psi\langle\tau_1, \ldots, \tau_m\rangle$ is a term, too. We call $\tau_1, \ldots, \tau_m$ *actual parameters of new kind.*

Definition 3.1. A string Π of ALGOL 60-P basic symbols added by $\langle$ and $\rangle$ is called a *syntactical ALGOL 60-P-G program,* if there is a synthetical ALGOL 60-P program Π' such that Π results from Π' in the following way:

a) Procedure headings in Π' are replaced by new headings of the form $(*)$ above.

b) Applied identifier occurrences are replaced by terms.

If in a syntactical program Π every occurrence (i, Z_i) of an identifier has a uniquely associated defining occurrence $\delta(i, Z_i)$ of this identifier, then Π is called a *formal* ALGOL 60-P-G program.

Modes are assigned to constants and identifier occurrences as in ALGOL 60-P. The mode of formal parameters of old and new kind is **formal** and the mode ∂f of the procedure identifier f of $(*)$ is

$$\mathbf{proc}\ \langle\underbrace{\mathbf{formal}, \ldots, \mathbf{formal}}_{m_f > 0 \text{ times}}\rangle\ (\underbrace{\mathbf{formal}, \ldots, \mathbf{formal}}_{n_f > 0 \text{ times}})$$

or

$$\mathbf{proc}\ \langle\underbrace{\mathbf{formal}, \ldots, \mathbf{formal}}_{m_f > 0 \text{ times}}\rangle, \qquad n_f = 0,$$

or

$$\mathbf{proc}\ (\underbrace{\mathbf{formal}, \ldots, \mathbf{formal}}_{n_f > 0 \text{ times}}), \qquad m_f = 0,$$

or

$$\mathbf{proc}\ 0, \qquad m_f = n_f = 0.$$

Modes of "applied occurrences" of terms τ are defined inductively:

1. If τ is an identifier then $\partial\tau$ is the mode of this identifier.

2. If τ is $\psi\langle\tau_1, \ldots, \tau_m\rangle$, $m \geqq 1$, then $\partial\tau$ is

$$\underbrace{\langle \textbf{formal}, \ldots, \textbf{formal}\rangle}_{m \text{ times.}}$$

Modes of right hand expressions in assignment statements or Boolean expressions in if clauses of conditional statements are defined as in ALGOL 60-P.

The definition of *compilable* ALGOL 60-P-G programs looks similar to that of ALGOL 60-P. Only condition 3) changes and splits to 3a), 3b), 3c):

3a) For any applied occurrence of a term $\psi\langle\tau_1, \ldots, \tau_m\rangle$, $m \geqq 1$, $\partial\psi\langle\tau_1, \ldots, \tau_m\rangle$ is a constituent of $\partial\psi$, i.e. the identifier ψ denotes a procedure with $m \geqq 1$ formal parameters of new kind.

3b) Any applied coccurrence of a procedure identifier ψ must be followed by a non-empty list $\langle\tau_1, \ldots, \tau_m\rangle$ of actual parameters of new kind if

$$\underbrace{\langle \textbf{formal}, \ldots, \textbf{formal}\rangle}_{m \geqq 1 \text{ times}}$$

is a constituent of $\partial\psi$.

3c) For any procedure statement

$$\begin{aligned} &\psi\,\langle\tau_1, \ldots, \tau_m\rangle\,(\alpha_1, \ldots, \alpha_n), \quad m, n > 0,\\ \text{resp. } &\psi\,\langle\tau_1, \ldots, \tau_m\rangle, \quad m > 0,\\ \text{resp. } &\psi\,(\alpha_1, \ldots, \alpha_n), \quad n > 0,\\ \text{resp. } &\psi \end{aligned}$$

where ψ is an identifier and $\tau_1, \ldots, \tau_m, \alpha_1, \ldots, \alpha_n$ are terms one of equations

$$\begin{aligned} &\partial\psi = \textbf{proc}\ \underbrace{\langle \textbf{formal}, \ldots, \textbf{formal}\rangle}_{m > 0 \text{ times}}\ \underbrace{(\textbf{formal}, \ldots, \textbf{formal})}_{n > 0 \text{ times}}\\ \text{resp. } &\partial\psi = \textbf{proc}\ \underbrace{\langle \textbf{formal}, \ldots, \textbf{formal}\rangle}_{m > 0 \text{ times}}\\ \text{resp. } &\partial\psi = \textbf{proc}\ \underbrace{(\textbf{formal}, \ldots, \textbf{formal})}_{n > 0 \text{ times}} \quad \text{or} \quad \partial\psi = \textbf{formal}\\ \text{resp. } &\partial\psi = \textbf{proc}\ 0 \qquad\qquad\qquad \text{or} \quad \partial\psi = \textbf{formal} \end{aligned}$$

holds. The properties to be a syntactical, formal, or compilable ALGOL 60-P-G program are decidable.

In the same way as for ALGOL 60-P we may introduce the notion of a *partially compilable* program and we may say when a program *Π' results from another program Π by application of the copy rule* ($\Pi \vdash \Pi'$).

Example:

```
Π¹ = begin proc f ⟨r⟩ (x); {r(x)};
           proc q(r); {q(f⟨r⟩); f⟨r⟩(r)};
           q(q) end
           ⊤
      ···{q(f⟨q⟩); f⟨q⟩(q)}···
           ⊤          ├──────────
···{q(f⟨f⟨q⟩⟩); f⟨f⟨q⟩⟩ (f⟨q⟩)}··· ···{q(q)}···
    ⊤              ⊤                     ⊤
    ⋮     ···{f⟨q⟩ (f⟨q⟩)}···             ⋮
                 ⊤
          ···{q(f⟨q⟩)}···
                 ⊤
                 ⋮
```

If $\Pi \overset{+}{\vdash} \Pi'$ then Π is partially compilable and Π' is at least formal. For an *original* program Π the *execution*

$$E_\Pi := \{\Pi' \mid \Pi \overset{*}{\vdash} \Pi'\}$$

and the *execution tree* $T_\Pi \subseteq E_\Pi$ are defined and it is clear when a program is called to have *formally correct parameter transmissions*. We may introduce the *reduced execution* $E_{r\Pi}$ and the *reduced execution tree* $T_{r\Pi}$. By their help we may say when two original programs are called *formally equivalent*. The property to have formally correct parameter transmissions is invariant for formally equivalent programs (see Theorem 4 in [4]).

The languages ALGOL 60-P and ALGOL 60-P-G are not essentially different if we restrict ourselves to programs without formal procedure calls. If Π is a compilable original ALGOL 60-P-G program of this form we replace in every procedure statement

$$\psi\langle\tau_1, \ldots, \tau_m\rangle(\alpha_1, \ldots, \alpha_n), \qquad m, n \geqq 0,$$

those actual parameters $\tau_1, \ldots, \tau_m, \alpha_1, \ldots, \alpha_n$ by ψ which are proper terms. In a further step we replace all opening brackets $\langle$ by (, all closing brackets $\rangle$ by), and all strings)(or $\rangle$(by , . We get a compilable original ALGOL 60-P program $\tilde{\Pi}$ which is formally equivalent with Π.

The difference between both languages lies in the method how procedures f which are called through formal procedure calls are supplied with actual parameters. In ALGOL 60-P all actual parameters are supplied at the moment of the call. In ALGOL 60-P-G actual parameters of old kind are supplied at the moment of the call whereas actual parameters of new kind may be considered as if they were supplied already at that moment when the procedure identifier f occurred as an actual parameter of a procedure statement.

Look at the following example

```
Π² = begin proc g; { };
           proc f⟨r⟩(x); { };
           proc q(y); {y(g)};
           q (f⟨q⟩) end
              ⊤
         ···{f⟨q⟩(g)}···
              ⊤
              ⋮
```

$f\langle q\rangle\,(g)$ is a call of the procedure f through the formal procedure call $y(g)$. The formal parameter x of f of old kind is replaced by g at the moment of this call. The formal parameter r of f of new kind was supplied already one step earlier in the procedure statement $q(f\langle q\rangle)$ where f occurred as an "actual parameter". In higher systems programming languages we may find such mechanisms which allow to supply a procedure with its actual parameters at different moments before the call.

If δf is the defining occurrence of a non-formal identifier f and if δg is the defining occurrence of an identifier g with a scope surrounding the scope of f then the relation $R_\Pi := \{(\delta f, \delta g)\}$ may be *extended* to all programs $\Pi' \in T_\Pi$ as in Definition 2.1 and 2.2. We may prove lemmata analogous to Lemmata 2.1 and 2.7. We should remark here that if g is formal then the images $R_{\Pi'}(\delta g')$ may be terms and not only defining occurrences of identifiers. A proper term $R_{\Pi'}(\delta g')$ or a non-formal identifier $R_{\Pi'}(\delta g')$ consists only of non-formal identifiers which are global to every procedure body in Π'. We say $R_{\Pi'}(\delta g')$ is *completely non-formal.* As we may assume programs to be distinguished we are allowed to identify defining occurrences of identifiers with the identifiers themselves.

As in 2. we are confronted with the problem to eliminate global procedure parameters. Let $\hat{\Pi}$ be a distinguished original program of the form

$$\textbf{begin} \ldots \underbrace{\textbf{proc}\; x_0\langle v_1, \ldots, v_{m_{x_0}}\rangle\,(x_1, \ldots, x_{n_{x_0}});\; \{\ldots a \ldots\}}_{\hat{\varphi}};\; \ldots\; \textbf{end}$$

We assume that in the body of $\hat{\varphi}$ there is at least one applied occurrence of an identifier a which is global to the extended body of $\hat{\varphi}$. a may be formal or non-formal. If a is a procedure identifier then the associated number m_a of formal parameters of new kind is assumed to be 0 and a is assumed to be different from x_0.

We alter $\hat{\Pi}$ to Π in the following way. $\hat{\varphi}$ gets an additional formal parameter $\bar{v}_0$ of new kind.

$$\textbf{proc}\; x_0\langle \bar{v}_0, v_1, \ldots, v_{m_{x_0}}\rangle\,(x_1, \ldots, x_{n_{x_0}});\; \{\ldots a \ldots\}.$$

Every applied occurrence of a term

$$x_0\langle \tau_1, \ldots, \tau_m\rangle$$

(where the parameterlist may be empty) is added by one additional actual parameter a of new kind

$$x_0\langle a, \tau_1, \ldots, \tau_m\rangle.$$

$\hat{\Pi}$ and Π are obviously formally equivalent. In $\Pi' \in T_\Pi$ there is no applied occurrence of any formal parameter $\bar{v}_0$.

Theorem 3.1. Let us rename some applied occurrences of a to $\bar{v}_0$ in the body of that procedure denoted by x_0. We get a new program $\tilde{\Pi}$. Then the distinguished, original programs Π and $\tilde{\Pi}$ are formally equivalent.

By Theorem 3.1 and Theorem 2.2 extended to ALGOL 60-P-G we get:

Theorem 3.2. There is an effective process which constructs for every original ALGOL 60-P program a formally equivalent ALGOL 60-P-G program without procedure nesting.

With the help of a generalized Theorem 3.1 which allows to eliminate a global procedure parameter which occurs simultaneously in several "parallel" procedures and with the help of a generalized Theorem 2.2 which is extended to ALGOL 60-P-G and which allows to move several "parallel" procedures simultanously we may conclude:

Theorem 3.3. There is an effective process which constructs for every original ALGOL 60-P-G program a formally equivalent one without procedure nesting.

The assumption $m_a = 0$ and $a \neq x_0$ if a is a procedure identifier forces us to apply the generalized Theorems. The proof principles of the generalized Theorems are the same as for Theorem 3.1 and 2.2.

Theorem 2.4, 2.6, 2.7, and 3.3 reveal a remarkable interpretation. If we define a program without procedure nesting to be a *modularly structured* program and if we define a programming language L to have the *modularity property* if every original program of L is formally equivalent to some modularly structured program of L then we can formulate (compare Dennis [2])

Theorem 3.4. ALGOL 60-P restricted to programs without global formal procedure parameters or without procedure identifiers as actual parameters and ALGOL 60-P-G have the modularity property. ALGOL 60-P does not have this property.

As large systems programs should be modularly structured we may conclude that ALGOL 60-P (and ALGOL 60) is not a best suitable systems programming language.

Let α be the declaration of a in Π. In order to prove Theorem 3.1 we consider the one-element relations

$$R_\Pi := \{(\delta x_0, \delta a)\}, \qquad R_{\tilde{\Pi}} := \{(\delta x_0, \delta a)\}$$

in Π and $\tilde{\Pi}$ and we extend these relations to all programs $\Pi' \in T_\Pi$ and $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ by Definition 2.1 or 2.2.

Lemma 3.1. For any term

$$u_0 \langle \tau_1, \ldots, \tau_m \rangle, \qquad m \geqq 0$$

in $\tilde{\Pi}'$ the following condition holds: If u_0 is the identifier x_0' of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$ then $m \geqq 1$ and τ_1 is either a term with $\delta u_0 R_{\Pi'} \tau_1$ or the first formal parameter $\bar{v}_0'$ of new kind of $u_0 (u_0 F \tau_1)$. $\tilde{\varphi}$ is the procedure declaration of x_0 in $\tilde{\Pi}$.

Proof. Lemma 3.1 is evident for $\tilde{\Pi}$. Let now $\tilde{\Pi}''$ be a generated program in $T_{\tilde{\Pi}}$ and we assume that Lemma 3.1 is already proved for the immediate predecessor $\tilde{\Pi}' \vdash \tilde{\Pi}''$. Let $\tilde{\Pi}''$ result form $\tilde{\Pi}'$ by the procedure statement

$$u_0 \langle \tau_1, \ldots, \tau_m \rangle \, (\alpha_1, \ldots, \alpha_n)$$

in Π'. If there is a term

$$u_0'' \langle \tau_1'', \ldots, \tau_{m''}'' \rangle, \qquad m'' \geqq 0$$

in the new modified body in $\tilde{\Pi}''$ then there is standing on the "corresponding" place in the body of u_0 a formal parameter

$$u_0'$$

of u_0 or a term

$$u_0' \langle \tau_1', \dots, \tau_{m''}' \rangle$$

where u_0'' is equal u_0' if u_0' is global to the body of u_0 or u_0'' is a modification of u_0' modified by the copy rule if u_0' is local.

In the first case $u_0'' \langle \tau_1'', \dots, \tau_{m''}'' \rangle$ is one of the actual parameters $\tau_1, \dots, \tau_m$, $\alpha_1, \dots, \alpha_n$ which are completely non-formal. By induction hypothesis and definition of $R_{\tilde{\Pi}''}$ we conclude the following: If u_0'' is the identifier x_0' of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$ then $m'' \geqq 1$ and τ_1'' is a term with $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$.

Let in the second case u_0' not denote any copy of $\tilde{\varphi}$. Then u_0'' does not denote any copy of $\tilde{\varphi}$, too, and we need not prove anything. Let now u_0' be the identifier x_0' of a copy $\tilde{\varphi}'$ of $\tilde{\varphi}$. Then $m'' \geqq 1$. We discuss the different situations a), b), c) in Definition 2.1 and a), b), c), d) in Definition 2.2.

a) u_0' is global to the body of u_0 and $u_0'' = u_0'$.

(1) If $\delta u_0' R_{\tilde{\Pi}'} \tau_1'$ then τ_1' is completely non-formal and global to the body of u_0 due to Lemma 2.1 or 2.7. Then $\tau_1' = \tau_1''$ and $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u_0' F \tau_1'$ then $u_0 = u_0'$ and $\tau_1'' = \tau_1$. τ_1 is completely non-formal with $\delta u_0 R_{\tilde{\Pi}'} \tau_1$. So $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

b) (1) If $\delta u_0' R_{\tilde{\Pi}'} \tau_1'$ then τ_1' is completely non-formal and global to the body of u_0. Then $\tau_1' = \tau_1''$.

(1.1) If u_0' is global, too, then $u_0'' = u_0'$ and $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(1.2) If u_0' is local, then u_0'' is the identifier x_0'' of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ and $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u_0' F \tau_1'$ then τ_1' is a formal parameter and u_0' and τ_1' are local to the body of u_0. Then u_0'' is the identifier x_0'' of a copy $\tilde{\varphi}''$ of $\tilde{\varphi}$ with $u_0'' F \tau_1''$.

c) in Definition 2.2:

(1) If $\delta u_0' R_{\tilde{\Pi}'} \tau_1'$ and

(1.1) if u_0' is global to the body of u_0 then τ_1' is global, too, with $u_0' = u_0''$ and $\tau_1' = \tau_1''$. We have $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(1.2) If u_0' is local then τ_1' is a formal parameter of u_0. τ_1'' is one of the actual parameters $\tau_1, \dots, \tau_m, \alpha_1, \dots, \alpha_n$ and is completely non-formal and global to the body of u_0. Then we have $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u_0' F \tau_1'$ then we argue as in case b) (2).

d) in Definition 2.2 and c) in Definition 2.1:

(1) If $\delta u_0' R_{\tilde{\Pi}'} \tau_1'$ and

(1.1) if u_0' is global to the body of u_0 we argue as in case c) (1.1).

(1.2) If u_0' is local then τ_1' is a local identifier, too. We have $\delta u_0'' R_{\tilde{\Pi}''} \tau_1''$ by definition of $R_{\tilde{\Pi}''}$.

(2) If $u_0' F \tau_1'$ then we argue as in case b) (2). Q.e.d.

Lemma 3.2. We consider T_Π and $T_{\tilde{\Pi}}$. Then for every program $\Pi' \in T_\Pi$ there exists exactly one ("nearly identical") program $\tilde{\Pi}' \in T_\Pi$ with the following properties:

1) Π' and $\tilde{\Pi}'$ are identical with the exception of some renamings of applied occurrences $\bar{v}_0'$ of identifiers of copies $\tilde{\xi}'$ of $\tilde{\xi}$ in $\tilde{\Pi}'$. $\tilde{\xi}$ is the declaration of the formal parameter $\bar{v}_0$ of new kind of x_0 in $\tilde{\Pi}$.

2) $\bar{v}_0'$ is renamed to an "applied occurrence" of a term τ' in Π'.

3) In Π' the relation

$$\pi\delta\bar{v}_0' R_{\Pi'} \tau'$$

holds. $\delta\bar{v}_0'$ is primarily a defining occurrence in $\tilde{\Pi}'$, but because of 1) also defining occurrence in Π'.

4) The relations $R_{\Pi'}$ and $R_{\tilde{\Pi}'}$ are identical.

Vice versa, for every program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$ there exists exactly one program $\Pi' \in T_\Pi$ with the same properties 1)–4).

Proof. For $\Pi' \in T_\Pi$ we have at most one $\tilde{\Pi} \in T_{\tilde{\Pi}}$. We consider $\Pi \in T_\Pi$, then $\tilde{\Pi} \in T_{\tilde{\Pi}}$ is a program with the desired properties.

Let now $\Pi'' \in T_\Pi$ be a generated program with the immediate predecessor $\Pi' \vdash \Pi''$. Let Π'' result from Π' by the procedure statement

$$u_0 \langle \tau_1, \ldots, \tau_m \rangle (\alpha_1, \ldots, \alpha_n)$$

in Π'. By induction hypothesis there exists a nearly identical program $\tilde{\Pi}' \in T_{\tilde{\Pi}}$. Since Π' is partially compilable $\tilde{\Pi}'$ is partially compilable, too, because of 1) and 2). In $\tilde{\Pi}'$ we have

$$u_0 \langle \tau_1, \ldots, \tau_m \rangle (\alpha_1, \ldots, \alpha_n)$$

on the "same" place as $u_0 \langle \tau_1, \ldots, \tau_m \rangle (\alpha_1, \ldots, \alpha_n)$ in Π' and this statement generates $\tilde{\Pi}'' \dashv \tilde{\Pi}'$. At first we can infer from 1)–4) for Π' and $\tilde{\Pi}'$ to 4) for Π'' and $\tilde{\Pi}''$.

We must now look at renamings in the bodies of those procedures in Π' and $\tilde{\Pi}'$ which are denoted by u_0. If we have such a renaming $\bar{v}_0'$ to τ' with $\pi\delta\bar{v}_0' R_{\Pi'} \tau'$ we discuss the situations a)–d) in the Definition 2.1 or 2.2 for $R_{\Pi''}$ and $R_{\tilde{\Pi}''}$.

a) Here u_0 is the identifier x_0' of copies φ' and $\tilde{\varphi}'$ of φ and $\tilde{\varphi}$ and $\bar{v}_0'$ is the first formal parameter of u_0. τ' is global to the body of u_0 and completely non-formal. τ' does not change when the copy rule is applied. $\bar{v}_0'$ is replaced by τ_1 when the copy rule is applied. By Lemma 3.1 $\delta u_0 R_{\Pi'} \tau_1$ holds. Because of $\pi\delta\bar{v}_0' R_{\Pi'} \tau'$ and due to Lemma 2.1 or 2.7 we have $\tau_1 = \tau'$, i.e. the renaming is cancelled.

b) Here $\pi\delta\bar{v}_0' = \delta x_0'$ is local to the body of u_0. τ' is completely non-formal and global to the body of u_0. τ' does not change when the copy rule is applied. $\bar{v}_0'$ is modified to $\bar{v}_0''$ when the copy rule is applied. $\pi\delta\bar{v}_0'' R_{\Pi''} \tau'$ holds by definition of $R_{\Pi''}$.

c) in Definition 2.2:

Here $\pi\delta\bar{v}_0' = \delta x_0'$ is local to the body of u_0 and τ' is a formal parameter of u_0. τ' is replaced by an actual parameter $\tau_1, \ldots, \tau_m, \alpha_1, \ldots, \alpha_n$ which is completely non-formal and global to the body of u_0. $\bar{v}_0'$ is modified to $\bar{v}_0''$. $\pi\delta\bar{v}_0'' R_{\Pi''}\tau''$ where τ'' is one of the actual parameters above holds by definition of $R_{\Pi''}$.

d) in Definition 2.2 and c) in Definition 2.1:

$\pi\delta\bar{v}_0' = \delta x_0'$ and τ' are local to the body of u_0. If a is formal, then τ' is a formal identifier, if a is non-formal then τ' is a non-formal identifier. $\bar{v}_0'$ and τ' are modified to $\bar{v}_0''$ and τ'' when the copy rule is applied. $\pi\delta\bar{v}_0'' R_{\Pi''}\tau''$ holds by definition of $R_{\Pi''}$.

The vice versa direction is similar. Q.e.d.

Theorem 3.1 is an immediate consequence of Lemma 3.2. We apply Theorem 3.1 to the example

```
Π̂³ = begin proc q (r);
              {proc f (x); {r (x)};
               q (f); f (r)};
        q (q) end
```

We get

```
Π̃³ = begin proc q (r);
              {proc f ⟨r⟩ (x); {r (x)};
               q (f ⟨r⟩); f ⟨r⟩ (r)};
        q (q) end
```

By Theorem 2.2 $\tilde{\Pi}^3$ is formally equivalent with Π^1 which we have become acquainted with before.

At this moment a "historical" remark seems to be useful in order to understand the proceeding of our investigations better. Our earliest conjecture was that the macro program problem for ALGOL 60-P is algorithmically solvable. Our first definite results were Corollary of Theorem 3 in [4] and Theorem 4.1 in the present paper. In order to reduce our conjecture to these results we needed an appropriate notion of equivalent programs such that the macro program property is invariant. So we introduced the notion of formally equivalent programs and we could actually prove Theorem 4 in [4] and Lemma 4.1 in the present paper. After this we were looking for an effective process which eliminates all global procedure parameters in a given program $\hat{\Pi}$, such that $\hat{\Pi}$ is formally equivalent with the transformed program $\tilde{\Pi}$. We defined and tried even more general processes than that process of Theorem 2.3 in the present paper. But all these processes failed because for each of them we found sample programs $\hat{\Pi}$ for which the transformed programs $\tilde{\Pi}$ were not formally equivalent with $\hat{\Pi}$. $\hat{\Pi}^3$ above is such a sample program. We found that infinitely many (!) accompanying formal parameters would be necessary. So we generalized ALGOL 60-P to ALGOL 60-P-G and we could prove Theorem 3.1–3.3. Unfortunately, we were not able to prove the solvability of the macro program problem for ALGOL 60-P-G programs without procedure nesting. So we dropped our earlier conjecture and replaced it by the contrary, namely Theorem 4.4.

In proofs on executions of programs procedure nestings are highly disturbing as applications of the copy rule yield additional procedure declarations (see e.g. Lemma 8 in [4]). The importance of ALGOL 60-P-G lies in the fact that we may restrict ourselves to programs without procedure nestings. So we are able to give a fairly short proof of the unsolvability of the macro program problem for ALGOL 60-P-G programs (Theorem 4.3). Beyond this, we were very lucky that the ALGOL 60-P-G programs $\Pi_{\mathfrak{K}}$ constructed in Lemma 4.3 are formally equivalent with the ALGOL 60-P programs $\tilde{\Pi}_{\mathfrak{K}}$ in Lemma 4.4. This proved our new conjecture (Theorem 4.4).

References

1. Aho, A. V.: Indexed grammars — an extension of the contextfree grammars. J. ACM **15**, 647–671 (1968)
2. Dennis, J. B.: Modularity. In: Bauer, F. L. (ed.): Advanced course on software engineering. Lecture Notes in Economics and Math. Syst. **81**, 128–182, Berlin-Heidelberg-New York: Springer 1973
3. Fischer, M. J.: Grammars with macro-like productions. Harvard University, Cambridge (Mass.), Report No. NSF-22. Math. Ling. and Autom. Translation, May 1968
4. Langmaack, H.: On correct procedure parameter transmission in higher programming languages. Acta Informatica **2**, 110–142 (1973)
5. Langmaack, H.: Über eine Beziehung zwischen ALGOL-Programmen und Makro-Grammatiken. In: Hotz, G., und Langmaack, H. (Hrsg.): Tagung über Automatentheorie und formale Sprachen, Oberwolfach 29. Okt.—4. Nov. 1972. Bonn: Mitteilungen der GMD Nr. 73/6, 1973
6. Naur, P. (ed.): Revised report on the algorithmic language ALGOL 60. Num. Math. **4**, 420–453 (1963)
7. Rounds, W. C.: Mappings and grammars on trees. Math. Systems Theory **4**, 257–287 (1970)

Prof. Dr. H. Langmaack
Fachbereich Angewandte Mathematik u. Informatik
der Universität des Saarlandes
D-6600 Saarbrücken
Bundesrepublik Deutschland