Jozef Gruska

Structural unambiguity of ALGOL MOD

Commentationes Mathematicae Universitatis Carolinae, Vol. 6 (1965), No. 3, 281--327

Persistent URL: http://dml.cz/dmlcz/105018

Commentationes Mathematicae Universitatis Carolinae
6, 3 (1965)

STRUCTURAL UNAMBIGUITY OF A L G O L M O D

J. GRUSKA, Bratislava

1. Introduction

The developing of the language ALGOL 60 was a very important event in the development of automatic programming.

The form and exactness by which the language ALGOL 60 was described (in contradistinction to description of languages developed until that date), the abundance and generality of its means of expression, the difficulty arising from the construction of translators, particularly effective translators, considerably stimulated the interest in progamming languages, especially in ALGOL 60, in problems of translation, and, moreover, in context-free grammars as one of the succesful means for language description.

In developing the language ALGOL 60 it was desired to create a machine independent language that would be suitable for the description of algorithms of numerical mathematics and approach as closely as possible the standard mathematic notation. Good readibility with little further explanation and the capacity of being translated into machine programs were to be the features of that language. Further, there had to be developed a language with such high degree of exactness of description that an algorithm described in this language and translated by means of translators designed for different

computers would yield the same result when used on different computers (with the exception of differences caused by different precision of arithmetic). Soon, however, it became obvious that this objective has not been attained. At the construction of translators it had been found that the description of that language is not as clear and precise as it has been assumed and needed. As a consequence of this authors of translators inclined to divergencies in explaining the description of ALGOL 60, [1] and one of the most important objectives - the uniformity of translation - was not achieved.

The use of context-free grammars for the description of syntax of ALGOL 60 not only essentially increased the interest in the study of these grammars but in connection with ALGOL 60 some new problems arose in this field: structural unambiguity of context-free grammars (see [5]) and a definition of semantics for languages the syntax of which is given as a context-free grammar.

The importance of structural unambiguity comes forward in connection with the language ALGOL 60. It has been pointed out by many authors that some ambiguities of the semantics of ALGOL 60 were consequences of the structural ambiguity of context-free grammar describing the syntax of ALGOL 60. (For example statement: *if* a *then* *for* i := 1 *while* n *do* *if* a *then* b *else* c .) It seems that the opinion that ALGOL 60 was structurally unambiguous was generally accepted when the Report [1] was published and also each time after a change was made to remove an ambiguity pointed out by an example. Thus, the experience with ALGOL 60 shows that it may happen that a context-free grammar, the structural unambiguity

of which is quite obvious, is not, in fact, structurally unambiguous.

The problem of the definition of semantics has been studied by V. Fabian, [7] for more general systems, called languages in his paper, than context-free grammars. He investigated such semantics S (a semantics is simply a transformation defined on the set of terminal texts derivable in a given language) that $S[A,t]$, for a text t derivable from a symbol A is determined, roughly speaking, by the way in which the text t is derivative from the symbol A, and showed that for such definition of semantics the structural unambiguity of a given language is very important. (More exactly the weak structural unambiguity.)

It would be very desirable if there could be given an algorithm for deciding for any context-free grammar, whether or not it is structurally unambiguous. Such an algorithm does not exist (D.G. Cantor [2], R.W. Floyd [8], N. Chomsky and M.P. Schützenberger [4]) even for very simple context-free grammar (A. Greibach [11]). Hence, for some grammars it can be very difficult to decide whether or not it is structurally unambiguous.

Of course, this does not mean that it is impossible to devise methods which may be useful to decide at least for some context-free grammars, whether or not they are structurally unambiguous. Some methods of this kind have recently been investigated by V. Fabian [7] and J. Gruska [12] for languages defined in [7].

The present paper is devoted to showing that the results of papers [7] and [12] have made it possible relatively simply

to prove the structural unambiguity of language ALGOL MOD (the definition of ALGOL MOD is given in Section 5), which is the slight modification of ALGOL 60.

In the following two sections we introduce basic notations, definitions and some results from papers [7,12,13,14] which will be useful in Section 6. In Section 4 the structural unambiguity of the language ALGOL 60 is investigated. The definition of ALGOL MOD is given in Section 5 and the proof of structural unambiguity of ALGOL MOD in Section 6.

2. Basic notations and definitions

In this section we introduce basic notations and definitions from Fabian's paper [7].

2.1. Sets. By $\{x; \mathcal{C}(x)\}$ we denote the set of all such x which satisfy condition $\mathcal{C}(x)$. $\{x; \mathcal{C}(x, \theta)\}$ is an abbreviation for $\{x$; there is a θ such that $\mathcal{C}(x, \theta)$ holds$\}$. Λ denotes the empty set.

2.2. Transformations. If F is a transformation, then by $\mathrm{d}F$ and $\mathrm{r}F$ we denote the domain and the range, respectively, of F ; by $F(x)$ or Fx - if there is no danger of misunderstanding - we denote the value of F at x. If F and G are transformations then FGx means $F(G(x))$. If $M \subset \mathrm{d}F$, then symbol F_M denotes the partial transformation on M.

2.3. Sequences. If t is a sequence (we shall consider only finite ones), then by $t(i)$ or ti we denote the i-th element of t. (Since t can be considered as the transformation defined on a set of integers.) λt is the

length of t, Λ denotes also the empty sequence.

By $[a_1, a_2, \dots, a_n]$, (or $a_1 a_2 \dots a_n$ - in examples) we denote the sequence t of the length n such that $ti = a_i$ for $1 \leqq i \leqq n$.

If t is a sequence, $1 \leqq i \leqq \lambda t$, $1 \leqq j \leqq \lambda t$, then by $t^{(i,j)}$ we denote the sequence of the length $j + 1 - i$ (or 0) if $j \geqq i$ (if $j < i$) such that $t^{(i,j)}(k) = t(i+k-1)$.

If M is a set of sequences, we define the sets

$$\underline{\text{symb}}_0\, M = \{t1;\ \lambda t \geqq 1,\ t \in M\}$$

$$\underline{\text{symb}}_\varepsilon\, M = \{t\lambda t;\ \lambda t \geqq 1,\ t \in M\}$$

$$\underline{\text{symb}}\, M = \{ti;\ t \in M,\ 1 \leqq i \leqq \lambda t\}$$

of all first symbols, last symbols and symbols, respectively, of sequences in M.

If M is a set we define the set

$$\aleph M = \{t;\ t \text{ is a sequence and } \underline{\text{symb}}\,\{t\} \subset M\}$$

of all sequences of the elements in M.

2.4. Operations with sequences and decompositions.

By $\times$ we denote the operation of concatenation of two sequences.

If τ is a sequence the elements of which are sequences, then by $\Pi\tau$ we denote the sequence $\tau 1 \times \tau 2 \times \dots \times \tau\lambda\tau$. τ is a decomposition if $\tau \in \aleph\aleph M$, i.e. if τ is a sequence of sequences. We say that τ is a decomposition of t if $\Pi\tau = t$.

Example 1. $\tau_1 = [abc, ce, ef]$ and $\tau_2 = [abc, \Lambda, ce, ef]$ are two decompositions of the sequence $t = abcceef$.

With every decomposition τ we associate the sequence $\iota\tau$ (so-called the index-decomposition of τ) of integers such that $\lambda(\iota\tau) = \lambda\tau + 1$ and $\tau i = (\Pi\tau)^{(\times i, \times(i+1))}$

if we denote $x = \iota\tau$.

Example 2. If τ_1 and τ_2 are as in Example 1, then $\iota\tau_1 = [1, 4, 6, 8]$, $\iota\tau_2 = [1, 4, 4, 6, 8]$.

If τ_1 and τ_2 are decompositions such that $\lambda\Pi\tau_1 = \lambda\tau_2$, then we define a new decomposition $\tau = \tau_1 \circledast \tau_2$ as the decomposition of the length $\lambda\tau_1$ such that $\tau i = \Pi\tau_2^{(x_1 i, x_1(i+1)-1)}$ where $x_1 = \iota\tau_1$.

Example 3. If $\tau_3 = [ab, c, de, f]$, $\tau_4 = [ab, e, g, ij, \Lambda, cd]$ then $\tau = \tau_3 \circledast \tau_4 = [abe, g; ij; cd]$.

2.5. Languages. The concept of language which is introduced in the following (what we call here a language may also be called a grammar of a language) is a generalization of the concept of context-free grammar of N. Chomsky [3]. A generalization consists in that the set of rules and the set of non-terminal symbols can be infinite.

2.5.1. Definition. (Def. 5.1,[7]) $\mathcal{L}$ is a language if $\mathcal{L}$ is a transformation, if there exists a set $\mathbb{A}$ such that

(1) $d\mathcal{L} \subset \mathbb{A}$, $r\mathcal{L} \subset \{M, \Lambda \neq M \subset s\,\mathbb{A}\}$

(2) $[A] \notin \mathcal{L}A$ x) for every $A \in d\mathcal{L}$.

Hence, for every $A \in d\mathcal{L}$, $\mathcal{L}A$ is a non-empty set of finite sequences (but it is possible that $\mathcal{L}A = \{\Lambda\}$, i.e., $\mathcal{L}A$ consists of one sequence- the empty sequence) the elements of which are from $\mathbb{A}$.

With a language $\mathcal{L}$ we associate the alphabet $a\mathcal{L}$ defined as the smallest set $\mathbb{A}$ for which (1) holds. The elements of $a\mathcal{L}$ will be called symbols of $\mathcal{L}$. The elements of the sets $d\mathcal{L}$, $a_t\mathcal{L} = a\mathcal{L} - d\mathcal{L}$, $\sigma\mathcal{L} = s\,a\mathcal{L}$, $\sigma_t\mathcal{L} = s\,a_t\mathcal{L}$ are called metasymbols (non-terminal symbols), terminal symbols, strings, terminal strings, respectively, of $\mathcal{L}$.

x) see the following page

We shall write $\mathcal{L} : [A] \Rightarrow b$ [xx] as an abbreviation for $A \in \mathrm{al}\,\mathcal{L}$, $b \in \mathcal{L}A$. The expression $[A] \Rightarrow b$ can be called the rule of language with the left side $[A]$ and the right side b. The concept of language can also be identified with the concept of a set (in general infinite) of rules such that no rule has the form $[A] \Rightarrow [A]$ (see condition (2)). The rules of the form $[A] \Rightarrow \Lambda$ and $[A] \Rightarrow [B]$ are permitted.

We write $\mathcal{L} : a \longrightarrow b$ if there are strings q_1, q_2, q_3 and a metasymbol A such that $a = q_1 \times [A] \times q_3$, $b = q_1 \times q_2 \times q_3$, $\mathcal{L} : [A] \Rightarrow q_2$. We say that σ is a derivation of a string t_2 from t_1 if σ is a sequence, $\lambda\sigma > 1$, $\sigma 1 = t_1$, $\sigma\lambda\sigma = t_2$ and $\sigma i \longrightarrow \sigma(i+1)$ for $i = 1, 2, \ldots \ldots \lambda\sigma - 1$. We write $t_1 \rightarrow t_2$ if there is a derivation of t_2 from t_1. We write $t_1 \Rightarrow t_2$ if either $t_1 = t_2$ or $t_1 \rightarrow t_2$.

If there is no danger of misunderstanding the symbol specifying the language will be deleted in $\Rightarrow$, $\rightarrow$. Now we define the sets

$$\mathrm{t}(\mathcal{L}, A) = \{ t ; [A] \rightarrow t \}$$

$$\mathrm{t}\,\mathcal{L} = \bigcup \{ \mathrm{t}(\mathcal{L}; A); A \in \mathrm{al}\,\mathcal{L} \}$$

$$\mathrm{t}_t\,\mathcal{L} = \mathrm{t}\,\mathcal{L} \cap \sigma_t\,\mathcal{L}$$

$$\{ t ; [A] \Rightarrow t \}$$

---

x) We distinguish between a symbol A and the sequence $[A]$ which has the length 1.

xx) The symbol $\Rightarrow$ and the symbols $\longrightarrow$, $\rightarrow$ introduced below are used in this paper in the same meaning as in [7] and hence in an other meaning than in some other papers (for example [3]).

of all A -texts, texts, terminal texts, r-texts (rule texts), respectively, of $\mathcal{L}$.

The fact that for every metasymbol A , the set of all rules of the form $[A] \Rightarrow b$ can be infinite, seems to be the most important in this generalization of the concept of context--free grammar. The consequence of this is that any set of strings can be generated by a language in our sense (it is sufficient to consider languages such that the set of metasymbols and the set of all r-texts which are not terminal, are finite), which is not true for context-free grammars and which can be important in some cases. Moreover, context-free grammars are used as the means for defining some sets of strings chiefly because by using these grammars it is possible to define simultaneously a structure of strings. If for some sets of strings the structure of strings is not too important (for example for identifiers, strings, comments in ALGOL 60), it might be simpler to define these sets otherwise. Moreover, finiteness of the set of rules seems to be irrelevant to the study of structural unambiguity. The generative capacity of the language is not the object of our study.

Example 4. Languages $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ are given by their set of rules.

$\mathcal{L}_1$: $A \Rightarrow BcAd$, $B \Rightarrow e$, $A \Rightarrow f$, $A \Rightarrow ecfd$, $A \Rightarrow FD$, $F \Rightarrow ec$, $D \Rightarrow fd$

$\mathcal{L}_2$: $A \Rightarrow BC$, $B \Rightarrow D$, $C \Rightarrow D$, $B \Rightarrow a$, $C \Rightarrow b$, $D \Rightarrow \Lambda$

$\mathcal{L}_3$: $A \Rightarrow Ab$, $A \Rightarrow Ac$, $A \Rightarrow a$

2.6. Grammatical elements. $[A, t]$ is a (terminal) grammatical element if A is a metasymbol and t is a (terminal) text such that $[A] \rightarrow t$. By $\mathcal{g}\mathcal{L}$ ($\mathcal{g}_t\mathcal{L}$) we denote the set of all grammatical elements (terminal grammatical elements) of the language $\mathcal{L}$.

2.7. Structures. We say that τ is an α-decomposition of t if τ is a decomposition of t, α is a string such that $\lambda\alpha = \lambda\tau$ and $[\alpha i] \Rightarrow \tau i$ for $1 \leqq i \leqq \lambda\alpha$.

We say that $[\alpha, \tau]$ is the structure of a grammatical element $[A, t]$ if one of the following conditions is satisfied.

(1) $[A] \Rightarrow \alpha \rightarrow t$, τ is an α-decomposition of t.

(2) $[A] \Rightarrow t$ and $\alpha = [A]$, $\tau = [t]$.

If a grammatical element $\mathcal{g} = [A, t]$ has a structure $[\alpha, \tau]$ such that (1) holds, then $[\alpha, \tau]$ characterizes in a certain meaning, the way by which t has been derivated from A (see Fig.1).

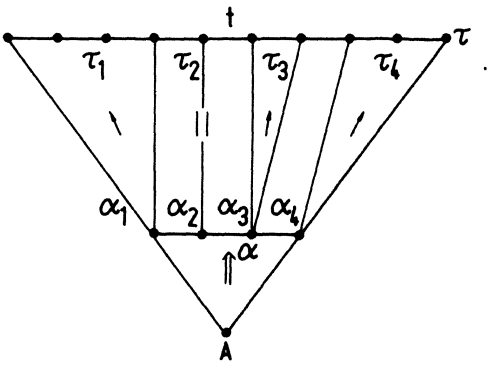

FIG.1

It seems to be convenient to define $[[A],[t]]$ as the structure of $[A,t]$ if $[A] \Rightarrow t$. (Of course, if $[A] \Rightarrow t$ then $[A,t]$ can have a structure $[\alpha,\tau]$ such that (1) holds.)

If g is a grammatical element of the language $\mathcal{L}$ then by $S_{\mathcal{L}} g$ ($\overline{S}_{\mathcal{L}} g$) we denote the set of all structures $[\alpha,\tau]$ (such that $\alpha \neq [A]$) of g and by Qg the set $\{g_1; [\alpha,\tau] \in \overline{S}_{\mathcal{L}} g, i \in d\alpha, g_1 = [\alpha i, \tau i] \in g\mathcal{L}\}$.

Example 5. Consider languages $\mathcal{L}_1$ and $\mathcal{L}_2$ from Example 4. The grammatical element $[A, ecfd] \in g\mathcal{L}_1$ has three structures: $[[A],[ecfd]]$, $[Bc\,Ad,[e,c,f,d]]$ and $[FD,[ec,fd]]$. The grammatical element $[A,D] \in g\mathcal{L}_2$ has two structures $[BC,[\Lambda,D]]$ and $[BC,[D,\Lambda]]$. For $g = [A, ecfd] \in g\mathcal{L}_1$ we have $Qg = \{[B,e],[A,f],[F,ec],[D,fd]\}$.

2.8. Structural unambiguity. Every grammatical element has a structure (Theorem 6.5, [7]). If a grammatical element g has exactly one structure, then g is said to be structurally unambiguous (shortly s.u.). If every (terminal) grammatical element of a language $\mathcal{L}$ is s.u., then $\mathcal{L}$ is said to be (weakly) structurally unambiguous.

Example 6. Languages $\mathcal{L}_1$ and $\mathcal{L}_2$ from Example 4 are not s.u.; $\mathcal{L}_3$ is s.u.. It is easy to see that $\mathcal{L}_2$ is weakly structurally unambiguous but not structurally unambiguous (see Example 5).

3. Structural unambiguity of languages

In papers [7,12] some necessary and sufficient conditions

for a language $\mathcal{L}$ to be s.u. have been proved. Many of these conditions are in such form that a given language $\mathcal{L}$ is s.u. if and only if so is a simpler language $\mathcal{L}_1$. In this section some of these results, which are needed in Section 6, are given.

3.1. Henceforth only such languages will be considered in which the following conditions are satisfied:

(1) $\mathcal{dl}\,\mathcal{L}$ and $\{\alpha;\ A \in \mathcal{dl}\,\mathcal{L}, [A] \Rightarrow \alpha \notin \mathfrak{S}_t\,\mathcal{L}\}$ are finite sets,

(2) $\mathcal{L}: t \rightarrow t$ for no string $t \in \mathfrak{S}\,\mathcal{L}$

although the results in paper [7] have been proved for languages without limitations (1) and (2).

The condition (1) means that the set of metasymbols is finite and for every metasymbol A there is only a finite number of rules $[A] \Rightarrow b$ such that b is not terminal string. However, the set of all rules can be infinite. If the condition (2) is not satisfied then $\mathcal{L}$ is not s.u. (Theorem 2.6,[14]).

3.2. Theorem. (Theorem 9.12,[7]) Let $\mathcal{L}$ be a language and a a set of metasymbols such that if $B \in \mathcal{dl}\,\mathcal{L} - a$, $[B] \Rightarrow t$ then $a \cap \underline{symb}\,\{t\} = \Lambda$. Let every $[A,t] \in \mathfrak{g}\,\mathcal{L}$ be s.u. if $A \in a$. Then $\mathcal{L}$ is s.u. if and only if so is $\mathcal{L}_{\mathcal{dl}\,\mathcal{L}-a}$. x)

One of the most important concepts of paper [7] is that of reducing transformation.

3.3. Definition. (See Def. 9.1,[7] and Theorem 2.12,[14]). Let there be transformations V, R and ρ defined on

---

x) See 2.2, i.e. $\mathcal{L}_{\mathcal{dl}\,\mathcal{L}-a}$ is a language such that $\mathcal{dl}\,\mathcal{L}_{\mathcal{dl}\,\mathcal{L}-a} = \mathcal{dl}\,\mathcal{L} - a$ and $\mathcal{L}_{\mathcal{dl}\,\mathcal{L}-a}(A) = \mathcal{L}\,A$.

$g\mathcal{L}$ such that for every $g = [A ; t] \in g\mathcal{L}$, every structure $[\alpha , \tau]$ of g we have:

(1) $\rho g = [A , Vg]$,

(2) Rg is a Vg -decomposition of t ,

(3) there is an α -decomposition ξ of Vg such that $\tau = \xi \circledast Rg$,

(4) if $[A] \Rightarrow t$, then $Vg \in \{[A] , t\}$,

(5) if $\rho g = g$, $g_1 \in Qg$ then $\rho g_1 = g_1$.

Then ρ is said to be a reducing transformation.

If $g = [A , t]$, then, by (2) and (3), $[A] \Rightarrow Vg \Rightarrow t$. Hence either $\rho g = [A, [A]]$ or $\rho g \in g\mathcal{L}$. If $Vg = [A]$, then, by (3), $[A] \Rightarrow t$ and the grammatical element $[A , t]$ is s.u. If $Vg \neq [A]$, then, with respect to (3), the grammatical element ρg has exactly so many structures as g .

The importance of reducing transformations follows from the following theorem.

3.4. Theorem. (Theorem 9.4, [7]) Let ρ be a reducing transformation for a language $\mathcal{L}$. Then $\mathcal{L}$ is s.u. if and only if every ρ -invariant grammatical element g (i.e. such that $\rho g = g$) is s.u..

Thus, investigating structural unambiguity it is not necessary to examine all the set $g\mathcal{L}$ but only $\{g ; \rho g = g\}$ where ρ is a reducing transformation.

A further important concept in paper [7] is that of isolable set.

3.5. Definition. (Def. 9.7, [7], Theorem 4.1, [12]) A nonempty set a of metasymbols is said to be isolable if there is a reducting transformation ρ such that

(1) if $\rho g = g, g_1 \in Qg, g_1 = [A, t]$, then $A \notin a$.

As a consequent of Theorem 9.13,[7] we have:

3.6. Theorem. Let $\mathcal{A}$ be an isolable set of metasymbols of $\mathcal{L}$ and let every $[A, t] \in \mathcal{G}\mathcal{L}$, $A \in \mathcal{A}$ be s.u. Then $\mathcal{L}$ is s.u. if and only if so is $\mathcal{L}_{\mathcal{A}\mathcal{L}-\mathcal{A}}$.

Now we shall investigate the relations between the structural unambiguity of a given language $\mathcal{L}$ and a new language $\mathcal{L}_0$ which is created from $\mathcal{L}$ in a way that in r-texts of $\mathcal{L}$ all symbols from a set $\mathcal{A}$ of metasymbols are replaced by new terminal symbols.

3.7. Definition. Let $\mathcal{L}$ be a language, $\mathcal{A}$ a set of metasymbols and φ a transformation such that

(1) $\mathcal{A}\varphi = \mathcal{A}\mathcal{L}$, $\varphi a = a$ if $a \notin \mathcal{A}$, $\varphi a \notin \mathcal{A}\mathcal{L}$ if $a \in \mathcal{A}$.

Denote by $\mathcal{L}_{\mathcal{A}}^{\varphi}$ the language defined as follows:

$$\mathcal{A}\mathcal{L}_{\mathcal{A}}^{\varphi} = \mathcal{A}\mathcal{L} \quad \text{and} \quad \mathcal{L}_{\mathcal{A}}^{\varphi} A = \{\overline{\varphi}\alpha;\ \alpha \in \mathcal{L} A\}$$

where

$$\overline{\varphi} t = \prod_{i=1}^{\lambda t} [\varphi t i] \quad \text{for every string } t.$$

A further important result for isolable sets is:

3.8. Theorem. (Theorem 9.11,[7]) If $\mathcal{A}$ is an isolable set of a language $\mathcal{L}$ and φ an one-to-one transformation satisfying (3.7.1), then $\mathcal{L}$ is s.u. if and only if so is $\mathcal{L}_{\mathcal{A}}^{\varphi}$.

3.9. Definition. Let φ be a transformation such that (3.7.1) holds and, moreover,

(1) $(A_1, A_2 \in \mathcal{A}) \wedge (\varphi A_1 = \varphi A_2)$ if and only if $\mathfrak{t}(\mathcal{L}, A_1) \cap \mathfrak{t}(\mathcal{L}, A_2) \notin \{\Lambda\}$.

If both languages $\mathcal{L}$ and $\mathcal{L}_{\mathcal{A}}^{\varphi}$ are s.u. or both are not s.u., then $\mathcal{A}$ is said to be a weakly isolable set.

Hence, if $\mathcal{A}$ is an (weakly) isolable set of metasymbols of a language $\mathcal{L}$, then $\mathcal{L}$ is s.u. if and only if so

is a simpler language. The rest of this section is devoted to the sufficient conditions for a set $\mathcal{A}$ to be isolable or weakly isolable. Here the concepts of recognizable and strongly recognizable set play an important rôle. Roughly speaking a set $\mathcal{A}$ of metasymbols is recognizable (or strongly recognizable) in a language $\mathcal{L}$ if it is possible to recognize in texts of the language the presence of inserted texts $t_0 \in \{ t ; [A] \to t, A \in \mathcal{A} \}$ (and their beginnings and ends) in a way that certain recursive properties are satisfied (see (3.10.2) and (3.11.2)).

3.10. Definition.[x] (Def.3.1,[12]) A subset $\mathcal{A} \subset \mathcal{dL}$ is said to be recognizable (f -recognizable) if there exists a function f such that $\mathcal{d}f \subset \mathcal{qL}$, $1 \leqq \leqq f[A,t] \leqq \lambda t$ for each $f[A,t] \in \mathcal{d}f$ and the following conditions are satisfied:

(1) If $A \in \mathcal{A}$, $[A] \to t$, then $[A,t] \in \mathcal{d}f$; if $[A] \Rightarrow t$, $[A,t] \in \mathcal{d}f$, then $A \in \mathcal{A}$.

(2) If

(2a) $[A,t] \in \mathcal{d}f$, $[\alpha,\tau] \in \bar{S}[A,t]$, $x = \iota\tau$, $xj \leqq f[A,t] < x(j+1)$

then

(2b) $[\alpha j] = \tau j$ implies $f[A,\alpha] = j$ and $[\alpha j_0, \tau j_0] \in \mathcal{d}f$ for no $1 \leqq j_0 \leqq \lambda\alpha$

(2c) $[\alpha j] \to \tau j$ implies $f[\alpha j, \tau j] = f[A,t] - xj + 1$.

---

x) The examples of recognizable and strongly recognizable sets are given in Section 6.

(3) If $g \in g\mathcal{L}$ and $\mathcal{Q}g \cap dlf \neq \Lambda$, then $g \in dlf$.

3.11. Definition. (Def. 3,2[12]) Let $\mathcal{Q}$ be an f-recognizable subset of $dl\mathcal{L}$. We shall say that the functions f_0 and f_1 indicate the beginning and the end for f if $dlf_0 = dlf = dlf_1$, $1 \leq f_0 g \leq fg \leq f_1 g \leq \lambda t$ for each $g = [A, t] \in dlf$ and the following conditions are satisfied:

(1) If $[A] \Rightarrow t$, then $f_0[A,t] = 1$, $f_1[A,t] = \lambda t$.

(2) If

(2a) $[A] \to t$, $[\alpha, \tau] \in \bar{S}[A,t]$, $x = l\tau$, $xj \leq f[A,t] < (j+1)$, then

(2b) $[\alpha j] = \tau j$ implies $f_0[A,t] = 1$, $f_1[A,t] = \lambda t$

(2c) $[\alpha j] \to \tau j$ implies $f_s[A,t] = f_s[\alpha j, \tau j] + xj - 1$

for $s = 0, 1$.

3.12. Definition. A subset $\mathcal{Q} \subset dl\mathcal{L}$ is said to be strongly recognizable if there exist functions f_0, f and f_1 such that $\mathcal{Q}$ is f-recognizable and the functions f_0 and f_1 indicate the beginning and the end for f. Now we define the sets of left and right delimiters for a set $\mathcal{Q} \subset dl\mathcal{L}$.

$ldel\ \mathcal{Q} = symb_c \{t; A \in dl\mathcal{L}, [A] \Rightarrow \alpha; 1 \leq i \leq \lambda\alpha, [\alpha i] \overset{\beta}{\Rightarrow}, \beta 1 \in \mathcal{Q}\}$;

$rdel\ \mathcal{Q} = symb_b \{t; A \in dl\mathcal{L}, [A] \Rightarrow \alpha, 1 \leq i \leq \lambda\alpha, [\alpha i] \overset{\beta}{\Rightarrow}, \beta\lambda\beta \in \mathcal{Q}; \alpha^{(1,i-1)} \Rightarrow t\}$

$\alpha^{(i+1,\lambda\alpha)} \Rightarrow t\}$.

The following lemma gives certain sufficient conditions for a set $\mathcal{Q}$ to be strongly recognizable.

Roughly speaking, it is the case that if for a text t and an i such that $1 \leq i \leq \lambda t$ we know that a substring $t^{(j,k)}$, $j \leq i \leq k$, was derivated from an $A \in \mathcal{Q}$ then either $k = \lambda t$ or k is determined

by the first pair of symbols such that the first of these symbols belongs to the set $symb_{\mathcal{L}}\, A$ and the second one is the right delimiter for A. Similarly for j:

3.13. Lemma. (Lemma 3.5, [12]) Let A be an f-recognizable subset of $dl\,\mathcal{L}$. Put $Q = \{\alpha i;\ i = f[A,\alpha],\ [A] \Rightarrow \alpha\}$; $\mathcal{L}_1 = \mathcal{L}_{dl\mathcal{L}\text{-}A}$ and, for each $q \in Q$, $\mathcal{B}_q = \{\alpha;\ [A] \Rightarrow \alpha,\ i = f[A,\alpha],\ \alpha i = q\}$, $A_q = \{A;\ [A] \Rightarrow \alpha \in \mathcal{B}_q\}$, $B_q = \{u;\ A \in A,\ [A] \Rightarrow \alpha,\ f[A,\alpha] = j,\ \alpha j = q;\ \mathcal{L}_1 : \alpha^{(1,j)} \Rightarrow u\}$, $E_q = \{u;\ A \in A,\ [A] \Rightarrow \alpha,\ f[A,\alpha] = j,\ \alpha j = q;\ \mathcal{L}_1 : \alpha^{(j,\lambda\alpha)} \Rightarrow u\}$.
Let $A \subset a_t\,\mathcal{L}$ and let the following conditions are satisfied for every $q \in Q$:

(1) If $u_1, u_2 \in E_q$ and $u_2 = u_1 \times t$, $t \neq \Lambda$ then $t\,1 \notin \underline{rdel}\ A_q$

(2) If $u_1, u_2 \in B_q$ and $u_2 = t \times u_1$, $t \neq \Lambda$ then $t \lambda t \notin \underline{ldel}\ A_q$

Then A is strongly recognizable.

The importance of strongly recognizable sets follows from the following result.

3.14. Theorem. (Theorem 4.4, [12]) Let A be a strongly recognizable subset of $dl\,\mathcal{L}$. Let, for every $A_1, A_2 \in A$, $A_1 \neq A_2$ implies $t(\mathcal{L}, A_1)\ t(\mathcal{L}, A_2) = \Lambda$. Then A is an isolable set.

The following lemma, very often used in Section 6, gives sufficient conditions for a set A to be strongly recognizable or weakly isolable in such a case that there is a set Q of terminal symbols such that $[A] \Rightarrow b$ is a rule of $\mathcal{L}$ and $bi \in Q$ if and only if $A \in A$, $i = 1$. Thus, it is easy to recognize the beginning of an inserted text

derivated from an $A \in \mathcal{A}$. Lemma 3.16 corresponds to the case in which it is easy to recognize the end of an inserted text and can be proved similarly as Lemma 3.15.

3.15. Lemma. (Lemma 5.7,[12]) Let $\Lambda \neq \mathcal{A} \subset \mathcal{dL}$, let $\Lambda \in \mathcal{L}A$ for no $A \in \mathcal{A}$, let $\mathcal{Q} = \{\alpha 1; \alpha \in \mathcal{L}A, A \in \mathcal{A}\} \subset \alpha_1 \mathcal{L}$,

(1) $A \in \mathcal{dL}, [A] \Rightarrow \alpha, \alpha i = q \in \mathcal{Q}$ implies $A \in \mathcal{A}$, $i = 1$.

Denote, for every $q \in \mathcal{Q}$, $\mathcal{B}_q = \{\alpha; [A] \Rightarrow \alpha, \alpha 1 = q\}$, $\mathcal{A}_q = \{A; [A] \Rightarrow \alpha \in \mathcal{B}_q\}$, $\mathcal{L}_1 = \mathcal{L}_{\mathcal{dL}-\mathcal{A}}$. If for each $q \in \mathcal{Q}$ at least one of the conditions:

(2) $\underline{symb}\{u^{(1,2u-1)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \cap \underline{symb}_\varepsilon\{u; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} = \Lambda$,

(3) $\underline{symb}\{u^{(2,2u)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \cap \underline{rdel}\ \mathcal{A}_q = \Lambda$

holds then $\mathcal{A}$ is a strongly recognizable set.

If for each $q \in \mathcal{Q}$ either conditions (2),(4) and (6) or conditions (3),(5) and (6) hold where

(4) $\underline{symb}_\varepsilon\{u; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \subset \alpha_1 \mathcal{L}$,

(5) if $A_1, A_2 \in \mathcal{A}$, $\mathit{t}(\mathcal{L}, A_1) \cap \mathit{t}(\mathcal{L}, A_2) \neq \Lambda$, $A_1 \in \underline{rdel}\ \mathcal{A}_q$, then $A_2 \notin \underline{symb}\{u^{(2,2u)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\}$;

(6) if $B \in \mathcal{dL}; \alpha_1, \alpha_2 \in \mathcal{L}B, \alpha_1 \neq \alpha_2, \lambda\alpha_1 = \lambda\alpha_2$ then there exists an i such that $1 \leq i \leq \lambda\alpha_1$ and either $\{\alpha_1 i, \alpha_2 i\} \notin \mathcal{A}$ or $\mathit{t}(\mathcal{L}, \alpha_1 i) \cap \mathit{t}(\mathcal{L}, \alpha_2 i) = \Lambda$.

Then $\mathcal{A}$ is a weakly isolable set.

3.16. Lemma. Let $\Lambda \neq \mathcal{A} \subset \mathcal{dL}$, let $\Lambda \notin \mathcal{L}A$ for $A \in \mathcal{A}$, let $\mathcal{Q} = \{\alpha\lambda\alpha; \alpha \in \mathcal{L}A, A \in \mathcal{A}\} \subset \alpha_\varepsilon \mathcal{L}$, let

(1) $A \in \mathcal{dL}, [A] \Rightarrow \alpha, \alpha i = q \in \mathcal{Q}$ implies $A \in \mathcal{A}$, $i = \lambda\alpha$.

Denote, for every $q \in \mathcal{Q}$, $\mathcal{B}_q = \{\alpha; [A] \Rightarrow \alpha, \alpha\lambda\alpha = q\}$, $\mathcal{A}_q = \{A; [A] \Rightarrow \alpha \in \mathcal{B}_q\}$, $\mathcal{L}_1 = \mathcal{L}_{\mathcal{dL}-\mathcal{A}}$. If for each $q \in \mathcal{Q}$ at least one of the conditions (2) and (3) holds, where:

(2) $\underline{symb}\{u^{(2,2u)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \cap \underline{symb}_b\{u; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} = \Lambda$

(3) $\underline{symb}\{u^{(1,2u-1)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \cap \underline{ldel}\ \mathcal{A}_q = \Lambda$.

Then $\mathcal{A}$ is a strongly recognizable set.

If for each $q \in Q$ either conditions (2),(4) and (3.15.6) or conditions (3),(5) and (3.15.6) hold where:

(4) $\underline{symb}_b\{u; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\} \subset \mathcal{A}_t\, \mathcal{L}$

(5) if $A_1, A_2 \in \mathcal{A}$, $t(\mathcal{L}, A_1) \cap t(\mathcal{L}, A_2) \neq \Lambda$, $A_1 \in \underline{ldel}\ \mathcal{A}_q$ then

$$A_2 \notin \underline{symb}\{u^{(1,2u-1)}; \mathcal{L}_1 : \alpha \Rightarrow u, \alpha \in \mathcal{B}_q\}\ ,$$

then $\mathcal{A}$ is a weakly isolable set.

3.17. Definition. A set $\mathcal{A} \subset \mathcal{A}\mathcal{L}$ is said to be paranthesized if there are two sets R and $L \subset \mathcal{A}_t\, \mathcal{L}$ such that

(1) $A \in \mathcal{A}\mathcal{L}$, $[A] \Rightarrow \alpha, \alpha i \in L (\in R)$ if and only if $A \in \mathcal{A}$, $i = 1 (= \lambda\alpha)$.

As a special case of Lemma 3.15 we have

3.18. Theorem. (Theorem 6.2,[12]) Let $\mathcal{A}$ be a paranthesized subset of $\mathcal{A}\mathcal{L}$ and let conditions (3.15.6) be satisfied. Then $\mathcal{A}$ is weakly isolable.

At last we give one result from paper [13].

3.19. Theorem. Let $\mathcal{L}_0, \mathcal{L}_1, \ldots \mathcal{L}_n$ be languages such that for every $i = 1, 2, \ldots, n$ there exist $A_i, \alpha_i, j_i, k_i, X_i$ such that

(1) $A_i \in \mathcal{A}\mathcal{L}_{i-1}$, $\alpha_i \in \mathcal{L}_{i-1} A_i$, $1 \leq j_i \leq k_i \leq \lambda\alpha_i$, $X_i \notin a\mathcal{L}_{i-1}$

and, moreover,

(2) $\mathcal{A}\mathcal{L}_i = \mathcal{A}\mathcal{L}_{i-1} \cup \{X_i\}$, $\mathcal{L}_i B = \mathcal{L}_{i-1} B$ if $B \notin \{A_i, X_i\}$, $\mathcal{L}_i A_i = (\mathcal{L}_{i-1} A_i - \{\alpha_i\}) \cup \{\alpha_i^{(1, j_i - 1)} \times [X_i] \times \alpha_i^{(k_i+1, \lambda\alpha_i)}\}$, $\mathcal{L}_i X_i = \{\alpha_i^{(j_i, k_i)}\}$.

Then the language $\mathcal{L}_n$ is said to be the extension of $\mathcal{L}_0$. Moreover, $\mathcal{L}_n$ is s.u. if and only if so is $\mathcal{L}_0$.

4. ALGOL 60 and the structural unambiguity

The language ALGOL 60 (speaking of it in the following we shall keep in mind the syntax of this language without the limitations given in the non-formal parts of [1]) is neither structurally unambiguous nor weakly structurally unambiguous. Some ambiguities have already been referred to by different authors and a few have been removed (see [15]). But even after these modifications ALGOL 60 is not structurally unambiguous.

Structurally ambiguous (s.a.) are grammatical elements $[\langle \text{open string} \rangle, t]$. For example a grammatical element $[\langle \text{open string} \rangle, a^n]$ [x] has $n+2$ different structures. Moreover, the grammatical element $[\langle \text{procedure heading} \rangle, \langle \text{empty} \rangle]$ is s.a. even then each terminal grammatical element $[\langle \text{procedure heading} \rangle, t]$ is s.u.. In both cases it is possible to remove this structural ambiguity by slight modification of ALGOL 60. From the point of view of the definition of semantics it is sufficient for a given language to be weakly structurally unambiguous and therefore it is not necessary to remove the ambiguity in $[\langle \text{procedure heading} \rangle, \langle \text{empty} \rangle]$.

The second group of structurally ambiguous grammatical elements deserves a more profound attention. To this group belong grammatical elements $[\langle \text{primary} \rangle, t]$, $[\langle \text{expression} \rangle, t_1]$, $[\langle \text{actual parameter} \rangle, t]$ where t is an $\langle \text{identifier} \rangle$ - text and t_1 an $\langle \text{expression} \rangle$ - text which does not contain symbols of arithmetic and logical operators. For example t_1 = *if* (*if* a *then* b *else* c) *then* a *else* (s).

---

x) by a^n we denote the sequence of length n, i-th-element of which is a.

In all these cases we have structurally ambiguous grammatical elements (for example the grammatical element $[\langle$ expression $\rangle, t_1]$ has structures $[[\langle$ arithmetic expression $\rangle], [t_1]]$, $[[\langle$ boolean expression $\rangle], [t_1]]$ and $[[\langle$ designational expression $\rangle], [t_1]]$) and, moreover, if $[\alpha_1, \tau_1]$ and $[\alpha_2, \tau_2]$ are two different structures, then $\alpha_1 \neq \alpha_2$, $\tau_1 = \tau_2$. In some cases it is possible "from a context to determine the only structure" but not always. For example, for the grammatical element $[\langle$ actual parameter $\rangle, [a]]$ it is sometimes possible only dynamically to determine whether a is an identifier of the variable of the real or the boolean type or an identifier of procedure etc. There are troubles with removing of structural ambiguity of this type due to the fact that identifier may denote elements of various character and that operators ( ) and *if then else* may have operands of various character (for example: arithmetic expressions, boolean expressions, statements and so on). It has been proved (see R.W. Floyd,[9]) that there is no language $\mathcal{L}$ which has a finite set of rules and such that for some $A \in \mathcal{A}\mathcal{L}$ the set $t_t(\mathcal{L}, A)$ contains all texts which "are obviously" ALGOL's programs and does not contain such texts which "obviously are not" ALGOL's programs (if we take into consideration the limitations given in non-formal parts of [1]).

As ALGOL 60 is neither structurally unambiguous nor weakly structurally unambiguous we have made changes in its syntax in order to obtain the structural unambiguous language which differs from ALGOL 60 as little as possible and contains all

ALGOL's texts. We have called the language obtained in this way ALGOL MOD and its definition is given in Section 5.

At the construction of ALGOL MOD the syntax of ALGOL 60 was modified as follows:

1. Comments are defined similarly as the other elements of the language. For this purpose new metasymbols ‹ begin ›, ‹ end ›, ‹ comment › and ‹ ; › are introduced.

2. We do not divide expressions into arithmetic expressions, boolean expressions and designational expressions. This is the main change and it agrees with Wright's proposal (see [16]) for a generalization of ALGOL.

3. In expressions *if* t *then* t_1 *else* t_2 for t any conditional expression can be given.

(These are the essential arrangements that change the set of terminal text generated by language ALGOL 60. Besides, some additional arrangements have been made but they have no influence on the set of terminal texts.)

4. We do not use the metasymbols denoting various types of identifiers (see K. Čulík [6]).

5. ‹ function designator › denotes only a function with parameters.

6. The modification of syntax for ‹ open string ›, ‹ unlabelled block ›, ‹ unlabelled compound › and for some other metasymbols.

5. ALGOL MOD.

In the next section the proof of structural unambiguity of language ALGOL MOD will be given. For this purpose we shall construct the sequence of languages $\mathcal{L}_0$ = ALGOL MOD,

$\mathcal{L}_1, \ldots, \mathcal{L}_{12}$ such that the language $\mathcal{L}_{i+1}$ is s.u. if and only if so is $\mathcal{L}_i$. The language $\mathcal{L}_{12}$ is already so simple that it is easy to verify its s.u. . Definition of ALGOL MOD is given below in such a form which enables us to see the analogy between ALGOL 60 and ALGOL MOD as well as possible. We shall use symbol :: = instead of $\Rightarrow$. Sequence of characters enclosed in the brackets $\langle \ldots \rangle$ represents, similarly as in ALGOL 60 the only metasymbol and this sequence will be chosen in a way to display the analogy with the corresponding metasymbol of ALGOL 60 (for example metasymbol $\langle$ form.par.part $\rangle$ corresponds to the metasymbol ‹formal parameter part › of the language ALGOL 60. Similarly as in ALGOL 60, underlining is used for defining independent terminal symbols, for example <u>begin</u> , <u>DE</u> . The formulas by which the sets $\mathcal{L}_0 A$, $A \in dl\, \mathcal{L}_0$ are defined (similarly as in [1] we shall speak about metalingvistic formulas) are preceded by some integers. The first integer denotes the current number of this formula and the others the current number of the given formula, or formulas which arose from the given formula at the extension x) of the language $\mathcal{L}_0$, in the sequence of formulas by which the language $\mathcal{L}_1$ is defined. Definition of the language $\mathcal{L}_1$ is given in the next section. This numeration is made in order to display the relation between languages $\mathcal{L}_0$ and $\mathcal{L}_1$, because in the sequence of formulas defining language $\mathcal{L}_1$ there is a different

---

x) See Theorem 3.19

ordering of formulas being discussed in the next section.

5.1. Definition of the language ALGOL MOD.

5.1.1. Basic concepts.

5.1.1.1. Basic symbols.

(1-53) ⟨letter⟩ ::= a | b | c | d | e | f | g | h | i | j | k | l | m |
n | o | p | q | r | s | t | u | v | w | x | y | z |
A | B | C | D | E | F | G | H | I | J | K | L | M |
N | O | P | Q | R | S | T | U | V | W | X | Y | Z

(2-54) ⟨digit⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

(3-55) ⟨log.value⟩ ::= *true* | *false*

(4-1) ⟨delimiter⟩ ::= ⟨operator⟩ | ⟨separator⟩ | ⟨bracket⟩ |
| ⟨declarator⟩ | ⟨specificator⟩

(5-2) ⟨operator⟩ ::= ⟨arith.op.⟩ | ⟨rel.op.⟩ | ⟨log.op.⟩ |
| ⟨seq.op.⟩

(6-3) ⟨arith.op.⟩ ::= + | − | × | / | ÷ | ↑

(7-56) ⟨rel.op.⟩ ::= < | ≦ | = | ≧ | > | ≠

(8-4) ⟨log.op.⟩ ::= ≡ | ⊃ | ∪ | ∩ | ¬

(9-5) ⟨seq.op.⟩ ::= *go to* | *if* | *then* | *else* | *for* | *do*

(10-6) ⟨separator⟩ ::= ∪ | . | $_{10}$ | : | ; | :=: | , | *step* | *until* |
| *while* | *comment*

(11-7) ⟨bracket⟩ ::= (|) | [|] | ⌜ | ⌝ | *begin* | *end*

(12-8) ⟨declarator⟩ ::= *own* | *boolean* | *integer* | *real* | *array* |
| *switch* | *procedure*

(13-9) ⟨specificator⟩ ::= *string* | *label* | *value*

(14-10) ⟨basic symbol⟩ ::= ⟨letter⟩ | ⟨digit⟩ | ⟨log.value⟩
| ⟨delimiter⟩

(15-69) $\mathcal{L}_o$ (⟨empty⟩) = {Λ} .

5.1.1.2. Comments.

(16-14) $\mathcal{L}_o$ ⟨ sequencel ⟩ = $\underline{\Phi}(a_t\, \mathcal{L}_o - \{\underline{end}, ;, \underline{else}\})^{x)}$

(17-15) $\mathcal{L}_o$ ⟨ seq.2 ⟩ = $\underline{\Phi}(a_t\, \mathcal{L}_o - \{ ; \})$

(18-16) $\mathcal{L}_o$ ⟨ seq.3 ⟩ = $\underline{\Phi}(a_t\, \mathcal{L}_o - \{', '\}) - \{\Lambda\}$

(19-11) ⟨ end ⟩ : : = *end* ⟨ sequencel ⟩

(20-12) ⟨ comment ⟩ : : = *comment* ⟨ seq.2 ⟩

(21-18) ⟨ ; ⟩ : : = ; | ; ⟨ comment ⟩ ⟨ ; ⟩

(22-19) ⟨ begin ⟩ : : = *begin* | *begin* ⟨ comment ⟩ ⟨ ; ⟩

5.1.1.3. Identifiers.

(23-70) ⟨ ident. ⟩ : : = ⟨ letter ⟩ | ⟨ ident. ⟩ ⟨ letter ⟩ |
| ⟨ ident. ⟩ ⟨ digit ⟩

5.1.1.4. Numbers.

(24-71) ⟨ uns.integer ⟩ : : = ⟨ digit ⟩ | ⟨ uns.integer ⟩ ⟨ digit ⟩

(25-72) ⟨ integer ⟩ : : = ⟨ uns.integer ⟩ | ⟨ add.op. ⟩ ⟨ uns.integer ⟩

(26-57) ⟨ decimal fraction ⟩ : : = . ⟨ uns.integer ⟩

(27-58) ⟨ exponent part ⟩ : : = $_{10}$ ⟨ integer ⟩

(28-73) ⟨ decimal number ⟩ : : = ⟨ uns.integer ⟩ | ⟨ decimal fraction ⟩ |
| ⟨ uns. integer ⟩ ⟨ decimal fraction ⟩

(29-74) ⟨ uns.number ⟩ : : = ⟨ decimal number ⟩ | ⟨ exponent part ⟩ |
| ⟨ decimal number ⟩ ⟨ exponent part ⟩

5.1.1.5. Strings.

(30-17) ⟨ open string ⟩ : : = ⟨ seq.3 ⟩ | ' ⟨ open string ⟩ ^ ⟨ open string ⟩ |
| ⟨ seq.3 ⟩ ^ ⟨ open string ⟩ ^ ⟨ open string ⟩ |
| ⟨ empty ⟩

---

x) Thus $\mathcal{L}_o$: ⟨ sequencel ⟩ : : = t if and only if t is a terminal string such that $t\, i \in \{\underline{end}, ;, \underline{else}\}$ for no i .

(31-13) ⟨string⟩ ::= ⌜⟨open string⟩⌝

5.1.2. Expressions.

5.1.2.1. Variables.

(32-35) ⟨subscr.list⟩ ::= ⟨expr.⟩ | ⟨subscr.list⟩, ⟨expr.⟩

(33-21-44) ⟨subscr.variable⟩ ::= ⟨ident.⟩ [⟨subscr.list⟩]

(34-75) ⟨variable⟩ ::= ⟨ident.⟩ | ⟨subscr.variable⟩

5.1.2.2. Function designators.

(35-37) ⟨act.param.⟩ ::= ⟨string⟩ | ⟨expr.⟩

(36-43) ⟨letter string⟩ ::= ⟨letter⟩ | ⟨letter string⟩
⟨letter⟩

(37-51-20) ⟨param.del.⟩ ::= ,|)⟨letter string⟩:(

(38-36) ⟨act.par.list⟩ ::= ⟨act.param.⟩ | ⟨act.par.list⟩ |
|⟨param.del.⟩ ⟨act.param.⟩

(39-22-45) ⟨f.design.⟩ ::= ⟨ident.⟩ (⟨act.par. list⟩)

5.1.2.3. Expressions.

(40-59) ⟨add.op.⟩ ::= + | −

(41-60) ⟨mult.op.⟩ ::= × | / | ÷

(42-114--30-50) ⟨primary⟩ ::= ⟨uns.number⟩ | ⟨variable⟩ | ⟨f.design.⟩ | |⟨log.value⟩ | (⟨expr.⟩)

(43-76) ⟨term-1⟩ ::= ⟨primary⟩ | ⟨term-1⟩ ↑ ⟨primary⟩

(44-77) ⟨term-2⟩ ::= ⟨term-1⟩ | ⟨term-2⟩ ⟨mult.op.⟩
⟨term-1⟩

(45-78) ⟨term-3⟩ ::= ⟨term-2⟩ | ⟨add.op.⟩ ⟨term-2⟩ |
|⟨term-3⟩ ⟨add.op.⟩ ⟨term-2⟩

(46-79) ⟨term-4⟩ ::= ⟨term-3⟩ ⟨rel.op.⟩ ⟨term-3⟩ |
|⟨term-3⟩

(47-80) ⟨term-5⟩ ::= ⟨term-4⟩ | ⌜⟨term-4⟩

(48-81) ⟨ term-6 ⟩ ::= ⟨ term-5 ⟩ | ⟨ term-6 ⟩ ∩ ⟨ term-5 ⟩

(49-82) ⟨ term-7 ⟩ ::= ⟨ term-6 ⟩ | ⟨ term-7 ⟩ ∪ ⟨ term-6 ⟩

(50-83) ⟨ term-8 ⟩ ::= ⟨ term-7 ⟩ | ⟨ term-8 ⟩ ⊃ ⟨ term-7 ⟩

(51-84) ⟨ term-9 ⟩ ::= ⟨ term-8 ⟩ | ⟨ term-9 ⟩ ≡ ⟨ term-8 ⟩

(52-46) ⟨ if clause ⟩ ::= *if* ⟨ expr. ⟩ *then*

(53-62-33-23) ⟨ expr. ⟩ ::= ⟨ term-9 ⟩ | ⟨ if clause ⟩ ⟨ expr. ⟩ *else* ⟨ expr. ⟩

(54-85) ⟨ label ⟩ ::= ⟨ ident. ⟩ | ⟨ uns.integer ⟩

5.1.3. Statements.

5.1.3.1. Compound statements and blocks.

(55-86) ⟨ unlab.b.stat. ⟩ ::= ⟨ assign.stat. ⟩ | ⟨ go to stat. ⟩ | ⟨ dummy stat. ⟩ | ⟨ procedure stat. ⟩

(56-87) ⟨ basic stat. ⟩ ::= ⟨ unlab.b.stat. ⟩ | ⟨ label ⟩ : ⟨ basic stat. ⟩

(57-88) ⟨ unc.stat. ⟩ ::= ⟨ basic stat. ⟩ | ⟨ compound stat. ⟩ | ⟨ block ⟩

(58-89) ⟨ statement ⟩ ::= ⟨ unc.stat. ⟩ | ⟨ cond.stat. ⟩ | ⟨ for stat. ⟩

(59-64-110) ⟨ stat.list ⟩ ::= ⟨ statement ⟩ | ⟨ stat.list. ⟩ ⟨ ; ⟩ ⟨ statement ⟩

(60-65-111) ⟨ declar.list ⟩ ::= ⟨ declar. ⟩ | ⟨ declar.list ⟩ ⟨ ; ⟩ ⟨ declar. ⟩

(61-90) ⟨ unlab.compound ⟩ ::= ⟨ begin ⟩ ⟨ stat.list ⟩ ⟨ end ⟩

(62-66-112) ⟨ unlab.block ⟩ ::= ⟨ begin ⟩ ⟨ declar.list ⟩ ⟨ ; ⟩ ⟨ stat.list ⟩ ⟨ end ⟩

(63-32) ⟨ compound stat. ⟩ ::= ⟨ unlab.compound ⟩ | ⟨ label ⟩ : ⟨ compound stat. ⟩

(64-31) ⟨ block ⟩ :: = ⟨ unlab.block ⟩ | ⟨ label ⟩ : ⟨ block ⟩

(65-91) ⟨ program ⟩ :: = ⟨ block ⟩ | ⟨ compound stat. ⟩

5.1.3.2. Assignment statements.

(66-92) ⟨ left part ⟩ :: = ⟨ variable ⟩ : =

(67-93) ⟨ l.p.list ⟩ :: = ⟨ left part ⟩ | ⟨ l.p.list ⟩
⟨ left part ⟩

(68-94) ⟨ assign.stat. ⟩ :: = ⟨ l.p.list ⟩ ⟨ expr. ⟩

5.1.3.3. Go to statements.

(69-52) ⟨ go to stat. ⟩ :: = *go to* ⟨ expr. ⟩

5.1.3.4. Dummy statements.

(70-95) ⟨ dummy stat. ⟩ :: = ⟨ empty ⟩

5.1.3.5. Conditional statements

(71-24) ⟨ if stat. ⟩ :: = ⟨ if clause ⟩ ⟨ unc.stat. ⟩

(72-63-26-34-25) ⟨ cond. stat. ⟩ :: = ⟨ if stat. ⟩ | ⟨ if clause ⟩
⟨ for stat. ⟩ | ⟨ if clause ⟩ ⟨ unc. stat. ⟩
else ⟨ statement ⟩ | ⟨ label ⟩ : ⟨ cond.stat. ⟩

5.1.3.6. For statements.

(73-38) ⟨ f.l.element ⟩ :: = ⟨ expr. ⟩ | ⟨ expr. ⟩ *step*
⟨ expr. ⟩ *until* ⟨ expr. ⟩ | ⟨ expr. ⟩ *while*
⟨ expr. ⟩

(74-39) ⟨ for list ⟩ :: = ⟨ f.l.element ⟩ | ⟨ for list ⟩ ,
⟨ f.l. element ⟩

(75-47) ⟨ for clause ⟩ :: = *for* ⟨ variable ⟩ : =
⟨ for list ⟩ *do*

(76-96-27) ⟨ for stat. ⟩ :: = ⟨ for clause ⟩ ⟨ statement ⟩ |
⟨ label ⟩ : ⟨ for stat. ⟩

5.1.3.7. Procedure statements.

(77-97) < procedure stat.> : : = < f.design.> | < ident.>

5.1.4. Declarations.

(78-98) < declar. > : : = < type decl.> | < array decl.> |
| < switch decl.> | < proc.decl. >

5.1.4.1. Type declarations.

(79-99) < type list > : : = < ident.> | < type list > , < ident.>

(80-61) < type > : : = *real* | *boolean* | *integer*

(81-100) < l.or o.type > : : = < type > | *own* < type >

(82-101) < type decl.> : : = < l.or o.type > < type list >

5.1.4.2. Array declarations.

(83-40) < bound pair > : : = < expr. > : < expr. >

(84-41) < b.p.list > : : = < bound pair > | < b.p.list > ,
< bound pair >

(85-28-48) < array segm. > : : = < ident.> [< b.p.list >] |
| < ident. > , < array segm. >

(86-102) < array list > : : = < array segm. > |
| < array list > , < array segm. >

(87-103) < array decl. > : : = *array* < array list > |
| < l.or o.type > *array* < array list >

5.1.4.3. Switch declarations

(88-104) < switch list > : : = < expr. > | < switch list > ,
< expr. >

(89-105) < switch decl. > : : = *switch* <ident.>: = <switch list>

5.1.4.4. Procedure declarations.

(90-42) < form.par.list > : : = < ident. > |
| < formpar.list > < param.del.> <ident.>

(91-29-49) ⟨ form.par.part ⟩ : : = ⟨ ident.⟩ | ⟨ ident.⟩
(⟨ form.par.list ⟩)

(92-106) ⟨ ident.list ⟩ : : = ⟨ ident.⟩ | ⟨ ident.list ⟩ , ⟨ ident.⟩

(93-107) ⟨ specifier ⟩ : : = *string* | ⟨type ⟩ | *array* | ⟨type ⟩ *array* |
| *label* | *switch* | *procedure*
| ⟨ type ⟩ *procedure*

(94-68) ⟨ specif.part ⟩ : : = *value* ⟨ ident.list ⟩ ⟨ ; ⟩ |
| ⟨ specifier ⟩ ⟨ ident.list ⟩ ⟨ ; ⟩ |
| ⟨ specif.part ⟩ ⟨ specifier ⟩
⟨ ident. list ⟩ ⟨ ; ⟩

(95-67-113) ⟨ proc.head. ⟩ : : = ⟨ form.par.part ⟩ ⟨ ; ⟩ ⟨specif.part⟩ |
| ⟨ form.par.part ⟩ ⟨ ; ⟩

(96-108) ⟨ proc.body ⟩ : : = ⟨ statement ⟩ | ⟨ code ⟩

(97-109) ⟨ proc.decl.⟩ : : = *procedure* ⟨proc.head.⟩
⟨ proc.body ⟩ | ⟨ type⟩ *procedure*
⟨ proc.head ⟩ ⟨ proc.body ⟩

6. Structural unambiguity of ALGOL MOD.

In the preceding section it has been mentioned that in proving the structural unambiguity of ALGOL MOD we shall construct the sequence of languages $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_{12}$. The sequence of metalingvistic formulas by which these languages are defined is given below. To the definition of language $\mathcal{L}_i$, $i = 1, 2, \dots, 8$ belongs that part of the given sequence which begins with the formula preceded by $: \mathcal{L}_i$ and ends with the formula preceded by the symbol $\mathcal{L}_i :$. For other languages, i.e. for $\mathcal{L}_9$, $\mathcal{L}_{10}$, $\mathcal{L}_{11}$, $\mathcal{L}_{12}$, the corresponding metalingvistic formulas form two seaparate sequences

designated similarly for both parts.

It is easy to see, from the definition of languages $\mathcal{L}_0$ and $\mathcal{L}_1$, that $\mathcal{L}_1$ is the extension of $\mathcal{L}_0$. At the extension those metalingvistic formulas of language $\mathcal{L}_0$ were changed which are marked on the left by an n-tuple of integers with $n \geq 2$. These integers, except the first, denote the formulas of language $\mathcal{L}_1$ which arose from the given formula. This designation enables us easily to determine how the extension has been made and which new metasymbols have been introduced.

Each formula by which the language $\mathcal{L}_1$ is defined is preceded by a pair of integers. The first one denotes the current number of the given formula and the second one the corresponding formula of language $\mathcal{L}_0$. In order to arrange a compact and clear inscription of languages $\mathcal{L}_1$ to $\mathcal{L}_{12}$, the order of metalingvistic formulas of language $\mathcal{L}_1$ differs from the order of formulas in the definition of $\mathcal{L}_0$.

6.1. Definition of languages $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{12}$.

: $\mathcal{L}_1$ (1-4) ⟨ delimiter ⟩ ::= ⟨ operator ⟩ | ⟨ separator ⟩ |
| ⟨ bracket ⟩ | ⟨ declarator ⟩ | ⟨ specificator ⟩

(2-5) ⟨ operator ⟩ ::= ⟨ arith.op. ⟩ | ⟨ rel. op. ⟩ |
| ⟨ log.op. ⟩ | ⟨ seq.op. ⟩

(3-6) ⟨ arith.op. ⟩ :: = + | - | × | / | ÷ | ↑

(4-8) ⟨ log.op. ⟩ :: = ≡ | ⊃ | ∪ | ∩ | ⌐

(5-9) ⟨ seq.op. ⟩ :: = go to | if | then | else | for | do |

(6-10) ⟨ separator ⟩ :: = ∪ | . | $_{10}$ | : | ; | := | , | step |
| until | while | comment

(7-11) ⟨ bracket ⟩ :: = (|) | [|] | ⌜ | ⌝ | begin | end

(8-12) ⟨ declarator ⟩ :: = own | boolean | integer | real |
| array | switch | procedure

(9-13) ‹ specificator › :: = *string* | *label* | *value*

(10-14) ‹ basic symbol › :: = ‹ letter › | ‹ digit › |
| ‹ log.value › | ‹ delimiter ›

: $\mathcal{L}_2$ (11-19) ‹ end › :: = *end* ‹ sequencel ›

(12-20) ‹ comment › :: = *comment* ‹ seq.2 ›

: $\mathcal{L}_3$ (13-31) ‹ string › :: = ‘ ‹ open string › ’

(14-16) $\mathcal{L}_1$ ‹ sequencel › = $\underline{\&}(a_t\, \mathcal{L}_1 - \{$*end*, ; , *else*$\})$

(15-17) $\mathcal{L}_1$ ‹ seq.2 › = $\underline{\&}(a_t\, \mathcal{L}_1 - \{;\})$

(16-18) $\mathcal{L}_1$ ‹ seq.3 › = $\underline{\&}(a_t\, \mathcal{L}_1 - \{‘, ’\}) - \{\wedge\}$

(17-30) ‹ open string › :: = ‹ seq.3 › | ‘ ‹ open string › ’
‹ open string › | ‹ empty › | ‹ seq.3 › ‘
‹ open string › ’ ‹ open string ›

: $\mathcal{L}_4$ (18-21) ‹ ; › :: = ; | ; ‹ comment › ‹ ; ›

(19-22) ‹ begin › :: = *begin* | *begin* ‹ comment › ‹; ›

: $\mathcal{L}_5$ (20-37) ‹ PAR.DEL › :: =) ‹ letter string › : (

: $\mathcal{L}_6$ (21-33) ‹ subscr.variable › :: = ‹ ident. › ‹ INDEX ›

(22-39) ‹ f.design. › :: = ‹ ident. › ‹ PARAM ›

(23-53) ‹ COND.EXPR.1 › :: = ‹ if clause › ‹ expr. › *else*

(24-71) ‹ if stat. › :: = ‹ if clause › ‹ unc.stat. ›

(25-72) ‹ B.COND.STAT.1 › :: = ‹ if clause ›
‹ unc.stat. › *else*

(26-72) ‹ FOR COND.STAT. › :: = ‹ if clause › ‹ for stat. ›

(27-76) ‹ UNL.FOR STAT. › :: = ‹ for clause ›
‹ statement ›

(28-85) ‹ array segm. › :: = ‹ ident. › ‹ SEGMENT › |
| ‹ ident. › , ‹ array segm. ›

(29-91) ‹ form.par.part › :: = ‹ ident. › | ‹ ident. ›
‹ FORM.P.P. ›

(30-42) ⟨PRIMAR⟩ ::= ⟨PRIM⟩

(31-64) ⟨block⟩ ::= ⟨unlab.block⟩ | ⟨label⟩ : ⟨block⟩

(32-63) ⟨compound stat.⟩ ::= ⟨unlab.compound⟩ |
| ⟨label⟩ : ⟨compound stat.⟩

: $\mathcal{L}_7$ (33-53) ⟨COND.EXPR.⟩ ::= ⟨COND.EXPR.1⟩ ⟨expr.⟩

(34-72) ⟨B.COND.STAT.⟩ ::= ⟨B.COND.STAT.1⟩
⟨statement⟩

: $\mathcal{L}_8$ (35-32) ⟨subscr.list⟩ ::= ⟨expr.⟩ | ⟨subscr.list⟩ ,
⟨expr.⟩

(36-38) ⟨act.par.list⟩ ::= ⟨act.param.⟩ |
|⟨act.par.list⟩ ⟨param.del.⟩ ⟨act.param.⟩

(37-35) ⟨act.param.⟩ ::= ⟨string⟩ | ⟨expr.⟩

(38-73) ⟨f.l.element⟩ ::= ⟨expr.⟩ |⟨expr.⟩ *step*
⟨expr.⟩ *until* ⟨expr.⟩ | ⟨expr.⟩ *while* ⟨expr.⟩

(39-74) ⟨for list⟩ ::= ⟨f.l.element⟩ | ⟨for list⟩ ,
⟨f.l.element⟩

(40-83) ⟨bound pair⟩ ::= ⟨expr.⟩ : ⟨expr.⟩

(41-84) ⟨b.p.list⟩ ::= ⟨bound pair⟩ | ⟨b.p.list⟩ ,
⟨bound pair⟩

(42-90) ⟨form.par.list⟩ ::= ⟨ident.⟩ | ⟨form.par.list⟩|
⟨param.del.⟩ ⟨ident.⟩

(43-36) ⟨letter string⟩ ::= ⟨letter⟩ | ⟨letter string⟩
⟨letter⟩

(44-33) ⟨INDEX⟩ ::= [⟨subscr.list⟩]

(45-39) ⟨PARAM⟩ ::= (⟨act.par.list⟩)

(46-52) ⟨if clause⟩ ::= *if* ⟨expr.⟩ *then*

(47-75) ⟨for clause⟩ ::= *for* ⟨variable⟩:=⟨for list⟩ *do*

(48-85) ⟨SEGMENT⟩ ::= [⟨b.p.list⟩]

(49-91) ⟨FORM.P.P.⟩ ::= (⟨form.par.list⟩)

(50-42) ⟨PRIM⟩ ::= (⟨expr.⟩)

(51-37) ⟨param.del.⟩ ::= ,|⟨PAR.DEL.⟩

: $\mathcal{L}_9$ (52-69) ⟨go to stat.⟩ ::= go to ⟨expr.⟩

(53-1) ⟨letter⟩ ::= a|b|c|d|e|f|g|h|i|j|k|
l	m	n	o	p	q	r	s	t	u	v
w	x	y	z	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O	P	R	S
Q	T	U	V	W	X	Y	Z

(54-2) ⟨digit⟩ ::= 0|1|2|3|4|5|6|7|8|9

(55-3) ⟨log.value⟩ ::= true | false

(56-7) ⟨rel.op.⟩ ::= < | ≦ | = | ≧ | > | ≠

(57-26) ⟨decimal fraction⟩ ::= .⟨uns.integer⟩

(58-27) ⟨exponent part⟩ ::= $_{10}$ ⟨integer⟩

(59-40) ⟨add.op.⟩ ::= + | −

(60-41) ⟨mult.op.⟩ ::= × | / | ÷

(61-80) ⟨type⟩ ::= real | integer | boolean

: $\mathcal{L}_{10}$ (62-53) ⟨expr.⟩ ::= ⟨term-9⟩ | ⟨COND.EXPR.⟩

(63-72) ⟨cond.stat.⟩ ::= ⟨if stat.⟩ | ⟨B.COND.STAT.⟩ |
|⟨FOR COND.STAT.⟩ | ⟨label⟩ : ⟨cond.stat.⟩

: $\mathcal{L}_{11}$ (64-59) ⟨stat.list⟩ ::= ⟨statement⟩ | ⟨STAT⟩ ⟨statement⟩

(65-60) ⟨declar.list.⟩ ::= ⟨declar.⟩ | ⟨DECL.⟩ ⟨declar.⟩

(66-62) ⟨unlab.block⟩ ::= ⟨begin⟩ ⟨UN.BL.1⟩⟨stat.list⟩
⟨end⟩

(67-95) ⟨proc.head.⟩ ::= ⟨PR.HEAD⟩ ⟨specif.part⟩ |
|⟨PR.HEAD⟩

(68-94) ⟨specif.part⟩ ::= value ⟨ident.list⟩ ⟨;⟩ |
|⟨specifier⟩ ⟨ident.list⟩⟨;⟩⟨specif.part⟩
⟨specifier⟩ ⟨ident.list⟩ ⟨;⟩

: $\mathcal{L}_{12}$ (69-15) $\mathcal{L}_1$ ⟨ empty ⟩ = { Λ }

(70-23) ⟨ ident. ⟩ : : = ⟨ letter ⟩ | ⟨ ident. ⟩ ⟨ letter ⟩ |

| ⟨ ident. ⟩ ⟨ digit ⟩

(71-24) ⟨ uns.integer ⟩ : : = ⟨ digit ⟩ | ⟨ uns.integer ⟩

⟨ digit ⟩

(72-25) ⟨ integer ⟩ : : = ⟨ uns.integer ⟩ | ⟨ add.op. ⟩

⟨ uns.integer ⟩

(73-28) ⟨ decimal number ⟩ : : = ⟨ uns.integer ⟩ |

| ⟨ decimal fraction ⟩ | ⟨ uns.integer ⟩

⟨ decimal fraction ⟩

(74-29) ⟨ uns.number ⟩ : : = ⟨ decimal number ⟩ |

| ⟨ exponent part ⟩ | ⟨ decimal number ⟩

⟨ exponent part ⟩

(75-34) ⟨ variable ⟩ : : = ⟨ ident. ⟩ | ⟨ subscr. variable ⟩

(76-43) ⟨ term-1 ⟩ : : = ⟨ primary ⟩ | ⟨ term-1 ⟩ ↑

⟨ primary ⟩

(77-44) ⟨ term-2 ⟩ : : = ⟨ term-1 ⟩ | ⟨ term-2 ⟩

⟨ mult.op. ⟩ ⟨ term-1 ⟩

(78-45) ⟨ term-3 ⟩ : : = ⟨ term-2 ⟩ | ⟨ add.op. ⟩ ⟨ term-2 ⟩ |

| ⟨ term-3 ⟩ ⟨ add.op. ⟩ ⟨ term-2 ⟩

(79-46) ⟨ term-4 ⟩ : : = ⟨ term-3 ⟩ ⟨ rel.op. ⟩ ⟨ term-3 ⟩ |

| ⟨ term-3 ⟩

(80-47) ⟨ term-5 ⟩ : : = ⟨ term-4 ⟩ | ¬ ⟨ term-4 ⟩

(81-48) ⟨ term-6 ⟩ : : = ⟨ term-5 ⟩ | ⟨ term-6 ⟩ ∩ ⟨ term-5 ⟩

(82-49) ⟨ term-7 ⟩ : : = ⟨ term-6 ⟩ | ⟨ term-7 ⟩ ∪ ⟨ term-6 ⟩

(83-50) ⟨ term-8 ⟩ : : = ⟨ term-7 ⟩ | ⟨ term-8 ⟩ ⊃ ⟨ term-7 ⟩

(84-51) ⟨ term-9 ⟩ : : = ⟨ term-8 ⟩ | ⟨ term-9 ⟩ ≡ ⟨ term-8 ⟩

(85-54) ⟨ label ⟩ : : = ⟨ ident. ⟩ | ⟨ uns.integer ⟩

(86-55) ⟨unlab.b.stat.⟩ ::= ⟨assign.stat.⟩ | ⟨go to stat.⟩ |
⟨dummy stat.⟩ | ⟨procedure stat.⟩

(87-56) ⟨basic stat.⟩ ::= ⟨unlab.b.stat.⟩ | ⟨label⟩ :
⟨basic stat.⟩

(88-57) ⟨unc.stat.⟩ ::= ⟨basic stat.⟩ | ⟨compound stat.⟩ |
⟨block⟩

(89-59) ⟨statement⟩ ::= ⟨unc.stat.⟩ | ⟨cond.stat.⟩ |
⟨for stat.⟩

(90-61) ⟨unlab.compound⟩ ::= ⟨begin⟩ ⟨stat.list⟩
⟨end⟩

(91-65) ⟨program⟩ ::= ⟨block⟩ | ⟨compound stat.⟩

(92-66) ⟨left part⟩ ::= ⟨variable⟩ :=

(93-67) ⟨l.p.list⟩ ::= ⟨left part⟩ | ⟨l.p.list⟩
⟨left part⟩

(94-68) ⟨assign.stat.⟩ ::= ⟨l.p.list⟩ ⟨expr.⟩

(95-70) ⟨dummy stat.⟩ ::= ⟨empty⟩

(96-76) ⟨for stat.⟩ ::= ⟨UNL.FOR.STAT.⟩ | ⟨label⟩ :
⟨for.stat.⟩

(97-77) ⟨procedure stat.⟩ ::= ⟨f.design.⟩ | ⟨ident.⟩

(98-78) ⟨declar.⟩ ::= ⟨type decl.⟩ | ⟨array decl.⟩ |
⟨switch decl.⟩ | ⟨proc. decl.⟩

(99-79) ⟨type list⟩ ::= ⟨ident.⟩ | ⟨type list⟩ ,
⟨ident.⟩

(100-81) ⟨l.or o.type⟩ ::= ⟨type⟩ | *own* ⟨type⟩

(101-82) ⟨type decl.⟩ ::= ⟨l.or o.type⟩ ⟨type list⟩

(102-86) ⟨array list⟩ ::= ⟨array segm.⟩ | ⟨array list⟩ ,
⟨array segm.⟩

(103-87) ⟨array decl.⟩ ::= *array* ⟨array list⟩ |
⟨l.or o.type⟩ *array* ⟨array list⟩

(104-88) ⟨ switch list ⟩ :: = ⟨ expr. ⟩ | ⟨ switch list ⟩ ,
⟨ expr. ⟩

(105-89) ⟨ switch decl. ⟩ :: = *switch* ⟨ ident. ⟩ : =
⟨ switch list ⟩

(106-92) ⟨ ident.list ⟩ :: = ⟨ ident. ⟩ | ⟨ ident.list ⟩ ,
⟨ ident. ⟩

(107-93) ⟨ specifier ⟩ :: = *string* | ⟨ type ⟩ | *array* | ⟨ type ⟩
array | *label* | *switch* | *procedure* |
| ⟨ type ⟩ *procedure*

(108-96) ⟨ proc.body ⟩ :: = ⟨ statement ⟩ | ⟨ code ⟩

(109-97) ⟨ proc.decl. ⟩ :: = *procedure* ⟨ proc.head ⟩
⟨ proc.body ⟩ | ⟨ type ⟩ *procedure* ⟨ pro c.head ⟩
⟨ proc.body ⟩

(110-59) ⟨ STAT ⟩ :: = ⟨ stat.list ⟩ ⟨ ; ⟩

(111-60) ⟨ DECL ⟩ :: = ⟨ declar.list ⟩ ⟨ ; ⟩

(112-62) ⟨ UN.BL.1 ⟩ :: = ⟨ d eclar.list ⟩ ⟨ ; ⟩

(113-95) ⟨ PR.HEAD ⟩ :: = ⟨ form.par.part ⟩ ⟨ ; ⟩

$\mathcal{L}_1$-$\mathcal{L}_6$: (114-42) ⟨ primary ⟩ :: = ⟨ uns.number ⟩ | ⟨ variable ⟩ |
| ⟨ f.design. ⟩ | ⟨ log.value ⟩ | ⟨ PRIMAR ⟩

⟨ COND.EXPR.1 ⟩ :: = *IF* ⟨ expr. ⟩ *else*

⟨ B.COND.STAT.1 ⟩ :: = *IF* ⟨ unc.stat. ⟩ *else*

: $\mathcal{L}_9$ ⟨ FOR COND.STAT ⟩ :: = *IF* ⟨ for stat. ⟩

⟨ if stat. ⟩ :: = *IF* ⟨ unc.stat. ⟩

⟨ UNL.FOR STAT. ⟩ :: = *FR* ⟨ statement ⟩

: $\mathcal{L}_{10}$, $\mathcal{L}_{11}$, $\mathcal{L}_{12}$ ⟨ subscr.variable ⟩ :: = ⟨ ident. ⟩ *IN*

⟨ f.design. ⟩ :: = ⟨ ident. ⟩ *PA*

⟨ array segm. ⟩ :: = ⟨ ident. ⟩ *SE* | ⟨ ident. ⟩ ,
⟨ array segm. ⟩

⟨ form.par.part ⟩ :: = ⟨ ident. ⟩ | ⟨ ident. ⟩ *PA*

< PRIMAR > :: = $\mathcal{PA}$

< block > :: = $\mathcal{UB}$ | < label > : < block >

$\mathcal{L}_7$: < compound stat. > :: = $\mathcal{UG}$ | < label > :

< compound stat. >

< COND.EXPR. > :: = $\mathcal{CE}$ < expr. >

$\mathcal{L}_8$, $\mathcal{L}_9$, $\mathcal{L}_{10}$: < B.COND.STAT. > :: = $\mathcal{CE}$ < statement >

< expr. > :: = < term-9 > | $\mathcal{CO}$

$\mathcal{L}_{11}$: < cond.stat. > :: = < if stat. > | $\mathcal{CO}$

< FOR COND.STAT. > | < label > : < cond.stat. >

< stat.list > :: = < statement > | $\mathcal{ST}$ < statement >

< declar.list > :: = < declar. > | $\mathcal{DE}$ < declar. >

< unlab.block > :: = < begin > $\mathcal{DE}$ < stat.list >

< end >

< proc.head > :: = $\mathcal{ST}$ $\mathcal{DE}$ | $\mathcal{ST}$

$\mathcal{L}_{12}$: < specif.part > :: = *value* < ident.list > <;> |

| < specifier > < ident.list > <;> |

| $\mathcal{DE}$ < specifier > < ident. list > <;>

- - - - - - - - - - - - -

It is obvious that all these languages satisfy condition (3.1.1) and it is easy to prove, by using the results of paper [14] that condition (3.1.2) is also satisfied.

6.2. In Section 6.4 it will be proved, for $i = 1, 2, \ldots, 11$ that, languages $\mathcal{L}_{i+1}$ and $\mathcal{L}_i$ are in one of the following three relations:

R1 $\mathcal{L}_{i+1} = \mathcal{L}_{i_{d\mathcal{L}_i - a_i}}$ where

R1.1 $a_i \subset d\mathcal{L}_i$, each grammatical element $[A,t]$ where $A \in a$ is s.u. and, moreover, if $B \in d\mathcal{L}_i - a_i$ and $[B] \Rightarrow t$ then *symb* $\{t\} \cap a = \Lambda$

R2 $\mathcal{L}_{i+1} = \mathcal{L}_{i \alpha \mathcal{L}_i - a_i}$ where

R2.1 a_i is a strongly recognizable set,

R2.2 each grammatical element $[A, t]$ where $A \in a$ is s.u. and if $A_1, A_2 \in a$, $A_1 \to t$, $A_2 \to t$ for a t, then $A_1 = A_2$.

R3 $\mathcal{L}_{i+1} = \mathcal{L}_{i a_i}^{\varphi_i}$ where

R3.1 a_i is a weakly isolable set,

R3.2 transformations φ_i satisfy conditions (3.7.1) and (3.9.1).

But this means that the language $\mathcal{L}_i$, $i = 1, 2, \ldots, 11$, is s.u. if and only if so is $\mathcal{L}_{i+1}$. Indeed, if languages $\mathcal{L}_{i+1}$ and $\mathcal{L}_i$ are in relation R1, then it follows from Theorem 3.2; if they are in relation R2, then from Theorem 3.14 and Theorem 3.6; if they are in relation R3, then it follows from the definition of weakly isolable set (see Def.3.9).

6.3. If, in the following, we shall want to prove that a language $\mathcal{L}_{i+1}$ is in relation R_j with $\mathcal{L}_i$, then we shall have to prove that

(1) condition Rj.1 (and Rj.2 if $j \in \{2, 3\}$, too) holds and, moreover, languages $\mathcal{L}_{i+1}$ and $\mathcal{L}_i$ are in the relation

(2) $\mathcal{L}_{i+1} = \mathcal{L}_{i \alpha \mathcal{L}_i - a_i}$ if $j \in \{1, 2\}$ and $\mathcal{L}_{i+1} = \mathcal{L}_{i a_i}^{\varphi_i}$ if $j = 3$ for a set a_i (and a transformation φ_i if $j = 3$).

If a_i and φ_i are given, then it is easy to verify, from the definition of languages $\mathcal{L}_i$ and $\mathcal{L}_{i+1}$, whether condition (2) is satisfied. In order to prove (1) we shall often use Lemma 3.13, Lemma 3.15, Lemma 3.16 and Theorem 3.18. But we shall not prove in detail that the assumptions of these lemmas and this theorem are satisfied, to the verification of which

it is sufficient to review the metalingvistic formulas of given language in finite number times, although it gives much labor in some cases (for example if it is necessary to determine the set $\underline{rdel}\ A$ for some $A \subset Al\ \mathcal{L}$). In such cases we shall either not speak about these assumptions at all or we shall say (as in the case of other assertions about languages which may be verified in this manner) that they follow from the definition of language $\mathcal{L}_i$ (shortly, from Di).

In order to show that the condition R2.1 holds we shall use either Lemma 3.15 or Lemma 3.16 or directly define a function f such that the set A_i is f -recognizable, verify conditions of Definition 3.10 and then prove, by using Lemma 3.13, that A_i is strongly recognizable.

For the proof of R 3.1 we shall use either Lemma 3.15 or Lemma 3.16 or Theorem 3.18.

In order to verify condition (3.15.6) we need to know from which metasymbols of A_i it is possible to derive the same text. For this purpose we shall write (wanting to show that languages $\mathcal{L}_{i+1}$ and $\mathcal{L}_i$ are in relation R3) the set A_i in the form $B_1 \cup B_2 \cup \ldots \cup B_n$ where $B_1, B_2, \ldots, B_n$ are disjoint sets and $A_1, A_2 \in A$, $t(\mathcal{L}_i, A_1) \cap t(\mathcal{L}_i, A_2) \subset \{\Lambda\}$ if and only if there is a j such that $A_1, A_2 \in B_j$. To make the verification of last condition easier we shall give so called characteristic sequence of texts for A_i , $[t_1, t_2, \ldots, t_k]$ (the arranging of $B_1, B_2, \ldots, B_n$ will be such that the sets $B_1, B_2, \ldots, B_k$ will have more than one element and the sets $B_{k+1}, \ldots, B_n$ will have just one element), such that $t_j \in t(\mathcal{L}_i, B)$ for any $B \in B_j$. In that case we shall define φ_i by n -tuple of symbols

$(X_1, X_2, \ldots, X_n)$ which will mean that $\varphi_i a = a$ if $a \notin A_i$ and $\varphi a = X_j$ if $a \in B_j$.

In all the cases A_i will be such that R1.1 and R2.2, respectively, follow from Di.

6.4. The proof of structural unambiguity of the language ALGOL MOD.

As already mentioned the language $\mathcal{L}_1$ is the extension of $\mathcal{L}_0$ and hence, by Theorem 3.19, $\mathcal{L}_1$ is s.u. if and only if so is $\mathcal{L}_0$. We extended the language $\mathcal{L}_0$ in a way to obtain the language in which, after removing some metasymbols (< sequencel >, < seq.2 >, < seq.3 >, < open string >, < PAR.DEL >) there already exists a parantheaized set (see set A_6 below; for this purpose "we have extended" the metalingvistic formulas 33,37,39,42,85 and 91 from the definition of language $\mathcal{L}_0$) and, moreover, it is possible to apply Lemmas 3.15 and 3.16.

By $\mathcal{L}_{i+1}$ ② $\mathcal{L}_i$ we shall indicate that the language $\mathcal{L}_{i+1}$ is in relation Rj with $\mathcal{L}_i$.

From D1 it is easy to see that the set

$A_1 = \{$ < delimiter >, < operator >, < arith.op. >, < log.op >, < seq.op. >, < separator >, < bracket >, < declarator >, < specificator >, basic symbol $\}$

satisfies R1.1 and, therefore, we can eliminate A_1 from $\mathcal{L}_1$. We obtain language $\mathcal{L}_2$ and it holds $\mathcal{L}_2$ ① $\mathcal{L}_1$. By eliminating the set

$A_2 = \{$ < comment >, < end >, < string > $\}$

from the language $\mathcal{L}_2$ we obtain language $\mathcal{L}_3$. Now we shall prove that $\mathcal{L}_3$ ② $\mathcal{L}_2$. Indeed, from D2 it is easy to see

that $\mathcal{A}_2$ satisfies R2.2. Now we prove that condition R2.1 is also satisfied. Put

$$Q = \{\underline{comment}, \underline{end}, \urcorner\}, \quad \mathcal{B}_2 = \{\langle \text{sequencel} \rangle, \langle \text{seq.2} \rangle, \langle \text{seq.3} \rangle, \langle \text{open string} \rangle\}.$$

Let χ be a function defined on $a\,\mathcal{L}_2$ in the following manner: $\chi(\ulcorner) = 1$, $\chi(\urcorner) = -1$ and $\chi(a) = 0$ if $a \in a\,\mathcal{L}_2 - \{\ulcorner, \urcorner\}$. Let

$$\bar{\chi}\,t = \sum_{i=1}^{\lambda t} \chi(ti) \text{ for any } t \in \mathcal{L}_2 .$$

By using structural induction (see Theorem 6.7, [7]) it is easy to prove:

(1) If $[\langle \text{string} \rangle] \to t$, then $\bar{\chi}\,t = 0$ and $\bar{\chi}\,t^{(1,i)} > 0$ if $1 \leqq i < \lambda t$.

From D2 it follows:

(2) $\underline{rdel}\ \{\langle \text{comment} \rangle\} = \{;, \langle ; \rangle\}$,

$\{;, \langle ; \rangle\} \cap \underline{symb}\{t, [\langle \text{comment} \rangle] \Rightarrow t\} = \Lambda$

(3) $\underline{rdel}\ \{\langle \text{end} \rangle\} = \{\langle \text{end} \rangle, \underline{end}, ;\langle ; \rangle, \underline{else}\} = N$,

$N \cap \underline{symb}\{t^{(2,\lambda t)}; [\langle \text{end} \rangle] \Rightarrow t\} = \Lambda$.

Let us define the function f as follows:

(4) $df = \{[A,t]; A \notin \mathcal{B}_2, i \in dt, ti \in Q\}, f[A,t] = min\{i; ti \in Q\}$.

Now we shall prove that all conditions of Definition 3.10 are satisfied, i.e. $\mathcal{A}_2$ is an f-recognizable set. (3.10.1) follows from D2. Now let (3.10.2a) hold. If $[\alpha j] = \tau j$, then $\alpha j \in Q$ and from D2 it follows: $A \in \mathcal{A}_2$, $f[A,\alpha] = 1 = j$, $[\alpha i, \tau i] \notin df$ if $i \in d\alpha$. Thus (3.10.2b) holds. If $[\alpha j] \to \tau j$, then from D2 it follows $A \notin \mathcal{A}_2 \cup \mathcal{B}_2$ and $\alpha j \notin \mathcal{B}_2$. Now (3.10.2c) follows from (4). For the proof that $\mathcal{A}_2$ is an f-recognizable set we must still show that (3.10.3) holds. Let $g \in g\,\mathcal{L}_2$, $g_1 \in Q\,g$. If $g_1 \in df$,

then $g_1 1 \notin \mathcal{B}_2$ according to (4) and either $g1 \in \mathcal{B}_2$ or $g \in dl\, f$. But from D2 it follows $g_1 1 \in \mathcal{B}_2$ if $g1 \in \mathcal{B}_2$. Hence and by previous $g \in dl\, f$.

In proving that $\mathcal{A}_2$ is strongly recognizable we shall use Lemma 3.13.(3.13.2) holds trivially since $B_q = \{q\}$ for each $q \in Q$ and (3.13.1) holds according to (1),(2) and (3). Thus, $\mathcal{A}_2$ is a strongly recognizable set and R2.1 holds.

In the language $\mathcal{L}_3$ the set

$\mathcal{A}_3 = \{ \langle \text{sequence1} \rangle , \langle \text{seq.2} \rangle , \langle \text{seq.3} \rangle , \langle \text{open string} \rangle \}$

satisfies R1.1 and therefore we can eliminate $\mathcal{A}_3$ from $\mathcal{L}_3$. We obtain exactly the language $\mathcal{L}_4$ which has the finite set of rules, and it holds $\mathcal{L}_4$ ① $\mathcal{L}_3$. If we eliminate from $\mathcal{L}_4$ the set

$\mathcal{A}_4 = \{ \langle \text{begin} \rangle , \langle ; \rangle \}$

we obtain the language $\mathcal{L}_5$. From D4 it follows that the set $\mathcal{A}_4$ satisfies R2.2. Now we prove that condition R2.1 holds also and hence $\mathcal{L}_5$ ② $\mathcal{L}_4$.

Put $Q = \{ ; , \underline{begin} \}$ and let us define a function f as follows:

$dl\, f = \{[A,t]; [A,t] \in g\mathcal{L}_5\, ti \in Q\}$, $f[A,t] = max\{i; ti \in Q\}$.

Similarly,as in proving Lemma 3.15 (i.e. Lemma 5.7,[12]) we can prove that the set $\mathcal{A}_4$ is f-recognizable. Since $\langle \text{comment} \rangle \notin \underline{rdel}\, \{\langle ; \rangle, \underline{begin}\}$,it is easy to see, by D4, that all assumptions of Lemma 3.13 are satisfied and therefore $\mathcal{A}_4$ satisfies condition R2.1.

The language $\mathcal{L}_6$ we obtain by eliminating the set

$\mathcal{A}_5 = \{ \langle \text{PAR.DEL} \rangle \}$

which, by D5, satisfies the condition R2.2. Now we prove that

condition R2.1 is also satisfied and, therefore, $\mathcal{L}_6 \textcircled{2} \mathcal{L}_5$. Put $Q = \{) \}$ and let us define a function f in the following manner:

$$dlf = \{[A,t]; i \in dlt, ti \in Q, t(i+1) \notin \underline{symb}_5\{t; [\langle \text{letter string} \rangle] \Rightarrow t\}) = N\}$$

$$f[A,t] = min\{i; ti \in Q,\ t(i+1) \in N\}$$

First we shall prove that the set $\mathcal{A}_5$ is f-recognizable, i.e. that conditions (3.10.1) to (3.10.3) of Lěmma 3.10 are satisfied. (3.10.1) follows from D5 and (3.10.3) from the definition of f . Now let (3.10.2a) hold. If $[\alpha j] = \tau j$, then, by D5, $A \in \mathcal{A}_5$, $j = 1 = f[A, \alpha]$ and (3.10.2b) holds. If $[\alpha j] \to \tau j$ and $f[A,t] < \times(j+1) - 1$ then (3.10.2c) holds trivially. From D5 we get that if $A \in dl\, \mathcal{L}_5$ and $) \in \underline{symb}_6\{t; [A] \Rightarrow t\}$ then $N \cap \underline{rdel}\{A\} = \Lambda$. Thus, it can not be $f[A,t] = \times(j+1)-1$ (3.10.2c) holds and $\mathcal{A}_5$ is f-recognizable set. Further it is obvious that all conditions of Lemma 3.13 are satisfied and hence $\mathcal{A}_5$ satisfies R2.1.

From D6 is is easy to see that the set

$$\mathcal{A}_6 = \{\langle \text{PARAM} \rangle, \langle \text{FORM.P.P.} \rangle, \langle \text{PRIM} \rangle\} \cup \{\langle \text{INDEX} \rangle\} \cup \{\langle \text{if clause} \rangle\} \cup$$
$$\cup \{\langle \text{for clause} \rangle\} \cup \{\langle \text{SEGMENT} \rangle\} \cup \{\langle \text{unlab.block} \rangle\} \cup$$
$$\cup \{\langle \text{unlab.compound} \rangle\}$$

is paranthesized and its characteristic sequence of texts is $[(\langle \text{ident.} \rangle]$ x). If we define $\mathcal{G}_6$ by 7-tuple $[\underline{PA}, \underline{IN}, \underline{IC}, \underline{FC}, \underline{SE}, \underline{UB}, \underline{UC}]$, then condition R3.2 is obviously satisfied and, by Theorem 3.18 and D6, R3.1 holds, too. It is easy to verify that $\mathcal{L}_7 = \mathcal{L}_6^{\mathcal{G}_6}{}_{\mathcal{A}_6}$ and hence $\mathcal{L}_7 \textcircled{3} \mathcal{L}_6$. $[\underline{IC}\ A\ \underline{else}\ ]$ is the characteristic text for the set

$$\mathcal{A}_7 = \{\langle \text{COND.EXPR.1} \rangle, \langle \text{B.COND.STAT.1} \rangle\}$$

in $\mathcal{L}_7$ and if we define $\mathcal{G}_7$ by symbol $\underline{CE}$ then, by D7, condition R3.2 holds. R3.1 follows from Lemma 3.16 (conditions

---

x) i.e. the characteristic sequence of texts has only one text.

(3.16.2) and (3.16.4) are satisfied) and from D7. Since $\mathcal{L}_{7_{a_7}}^{\varphi_7} = \mathcal{L}_8$ we have $\mathcal{L}_8$ ③ $\mathcal{L}_7$. In the language $\mathcal{L}_8$ the set

$a_8 = \{$ < subscr.list > , < act.par.list > , < act.param. > ,
< f.l.element > , < for list > , < bound pair > ,
< b.p.list > , < form.par.list > , < letter string > ,
< INDEX > , < PARAM > , < if clause > , < SEGMENT > ,
< FORM.P.P. > , < PRIM > , < param.del.>, < COND.EXPR.1 >,
< B.COND.STAT.1 > $\}$

satisfies condition R1.1 and can be eliminated from $\mathcal{L}_8$. We obtain the language $\mathcal{L}_9$ and it holds $\mathcal{L}_9$ ① $\mathcal{L}_8$. In this language the set

$a_9 = \{$ < go to stat.> , < letter > , < digit > , < log.value >,
< rel.op.> , < decimal fraction > , < exponent part >,
< add.op. >, < mult.op. > , < type > , < FOR COND.STAT.> ,
< if stat.> , <UNL.FOR STAT. > $\}$

satisfies, by D9, condition R2.2 and, by Lemma 3.15 (conditions (3.15.3) and (3.15.5) hold) condition R2.1, too. Hence we can eliminate a_9 from $\mathcal{L}_9$. Since $\mathcal{L}_{10} = \mathcal{L}_{9_{\mathcal{U}\mathcal{L}_9 - a_9}}$, we have $\mathcal{L}_{10}$ ② $\mathcal{L}_9$. The text $\underline{\mathcal{CE}}$ < ident. > is the characteristic one for the set

$a_{10} = \{$ < COND.EXPR. > , < B.COND.STAT > $\}$.

Thus, if we define φ_{10} by symbol $\underline{\mathcal{CO}}$, then from D10 and from Lemma 3.15 it follows (conditions (3.15.3) and (3.15.5) are satisfied) that the set a_{10} and transformation φ_{10} satisfy conditions R3.1 and R3.2. Moreover, $\mathcal{L}_{11} = \mathcal{L}_{10}^{\varphi_{10}}{}_{a_{10}}$ and hence $\mathcal{L}_{11}$ ③ $\mathcal{L}_{10}$.

In the language $\mathcal{L}_{11}$, the set

$a_{11} = \{ \langle$ STAT $\rangle$, $\langle$ PR.HEAD $\rangle \} \cup \{ \langle$ DECL $\rangle$, $\langle$ UN.B.1 $\rangle$, $\langle$ specif.part $\rangle \}$

has the characteristic pair of texts $[\langle$ ident. $\rangle \langle ; \rangle$, **procedure** $\langle$ ident. $\rangle \langle ; \rangle]$.

If we define g_{11} by ($\mathcal{LT}$, $\mathcal{DE}$), then, by Lemma 3.16 (conditions (3.16.3) and (3.16.5) are satisfied), the set a_{11} and transformations g_{11} satisfy conditions R3.1 and R3.2. Moreover, $\mathcal{L}_{12} = \mathcal{L}_{11}^{g_{11}, a_{11}}$ and hence $\mathcal{L}_{12} \textcircled{3} \mathcal{L}_{11}$.

The language $\mathcal{L}_{12}$ is very simple (it is non-self-embedding context-free grammar (see [3]) and also sequential grammar (see [10]) and it is easy to prove, for each $A \in \mathcal{N} \mathcal{L}_{12}$, that all grammatical elements $[A, t]$ are s.u..

Since the language $\mathcal{L}_{12}$ is s.u. and a language $\mathcal{L}_i$, $i = 0, 1, \ldots, 11$, is s.u. if and only if so is $\mathcal{L}_{i+1}$, we get that the language $\mathcal{L}_0$ = ALGOL MOD is s.u..

6.5. At the investigation of the structural unambiguity of language $\mathcal{L}_{12}$ we can henceforth proceed as in Section 6.4. Now we show one of the possible methods. Put

$a_{12} = \{ \langle$ ident. $\rangle$, $\langle$ uns.integer $\rangle \}$

$a_{13} = \{ \langle$ empty $\rangle$, $\langle$ integer $\rangle$, $\langle$ array segm. $\rangle$, $\langle$ form.par.part $\rangle$, $\langle$ block $\rangle$, $\langle$ compound stat. $\rangle$, $\langle$ stat.list $\rangle$, $\langle$ declar.list $\rangle$, $\langle$ COND.EXPR. $\rangle$, $\langle$ B.COND.STAT. $\rangle$, $\langle$ cond.stat. $\rangle$, $\langle$ unlab.block $\rangle$, $\langle$ proc.head $\rangle$, $\langle$ specif.part $\rangle \} \cup N$,

where N is the set of all metasymbols which are on the left side of formulas number 85 to 113.

$a_{14} = \{ \langle$ f.design $\rangle$, $\langle$ PRIMAR $\rangle$, $\langle$ subscr.variable $\rangle \}$

$a_{15} = \{ \langle$ variable $\rangle$, $\langle$ decimal number $\rangle \}$

$a_{16} = \{ \langle$ unsign. integer $\rangle \}$

Put $\mathcal{L}_{i+1} = \mathcal{L}_{i\,d\,\mathcal{L}_i - a_i}$ for $i = 12, 13, 14, 15, 16$.

It is easy to see that $\mathcal{L}_{14} \subseteq \mathcal{L}_{13}$. By using Lemma 3.15 and 3.16, we can prove that $\mathcal{L}_{i+1} \subseteq \mathcal{L}_i$ for $i = 12, 14, 15, 16$.

The structural unambiguity of language $\mathcal{L}_{17}$ follows from Example 10.9, [7]. Hence, the language $\mathcal{L}_{12}$ is s.u.

B i b l i o g r a p h y :

[1] J.V. BACKUS, F.L. BAUER, J. GREEN, C. KATZ, J.Mc CARTHY, P. NAUR (editor), A.J.PERLIS, H. RUTISHAUSER, K. SAMELSON, E. VAUQUOIS, J.H. WEGSTEIN, A.van WIJNGAARDEN, M. WOODGER: Report on the Algorithmic Language ALGOL 60, Numerische Mathematik 2(1960),106-136.

[2] D.G. CANTOR: On the ambiguity problem of Backus systems, J.Assoc.Comp.Mach. 9(1962),477-479.

[3] Noam CHOMSKY: On certain formal properties of grammars. Information and Control 2(1959),137-167.

[4] N. CHOMSKY and M.P. SCHÜTZENBERGER: The algebraic theory of contex-free languages, Computer Programming and formal systems.(Ed.P. BRADFFORT and D. HIRSCHBERG), Amsterdam 1963.

[5] N. CHOMSKY: Formal properties of grammars, Hadnbook of Mathematical Psychology, V 2, New York,Willey.

[6] Karel ČULÍK: Formal structure of ALGOL and simplification of its description, Symbolic languages in data processing, Rome 1962, 75-82.

[7] Václav FABIÁN: Structural unambiguity of formal languages, Czech.Math.J.14(89)(1964),394-430.

[8] R.W. FLOYD: On ambiguity in phrase structure of languages, Comm.ACM 5,10(Oct.1962),526-534.

[9] R.W. FLOYD: On the nonexistence of a phrase structure grammar for ALGOL 60. Comm.ACM,5,9(sept. 1962).

[10] Seymond GINSBURG and N. GORDON RICE: Two families of languages related to ALGOL, J.ACM 9(1962), 350-371.

[11] Sheila A. GREIBACH: The undecidability of the ambiguity problem for minimal linear grammars, Information and Control 6(1963),119-125.

[12] Jozef GRUSKA: Isolable and weakly isolable sets.(To appear in Czech.Math.Journal.)

[13] J. GRUSKA: Two operations with formal languages.(To appear)

[14] J. GRUSKA: On structural unambiguity of formal languages,Czech.Math.J. 15(90)(1965),283-294.

[15] REVISED REPORT ON ALGOL 60, Comm.ACM,5(1962),299-314.

[16] Niclaus WIRTH: A generalization of ALGOL, Comm.ACM V 6,N.9(1963),547-554.

(Received January 12, 1965)