RT-MAP-8208

UM GRAFO SINTÁTICO PARA A LINGUAGEM PL/M-80

V.W. Setzer

NOVEMBRO 1982

UM GRAFO SINTÁTICO PARA A LINGUAGEM PL/M-80

V.W. Setzer
Depto. de Matemática Aplicada
Instituto de Matemática e Estatística da USP

1. Introdução

A linguagem PASCAL foi definida por N. Wirth /J-W 74/ com a apresentação de um Grafo Sintático para a mesma. O tipo desse grafo é derivado de um grafo introduzido por Conway /CON 63/, e usado por Gries em seu conhecido livro sobre Compilação /GRI 71/. Wirth usou esse grafo com base para a programação de um analisador sintático do tipo Descendente-Recursivo /GRI 71, WIR 71, KOW 79/; no entanto, como mostramos em outro trabalho /S-H 81/, o grafo sintático apresentado em /J-W 74/ não se presta a uma programação direta para o método citado. Em /SET 79/, mostramos como se pode construir um algoritmo não-recursivo que interpreta uma tabela que representa um grafo sintático. Em /S-H 81/ damos formalmente restrições que devem ser seguidas pelo grafo para que corresponda a uma análise sintática correta, bem como introduzimos técnicas de recuperação automática de erros. Aquelas restrições servem também para definir a classe de gramáticas que denominamos de ESLL(1), aceitas pelo algoritmo de análise sintática. Denominaremos a classe de grafos sintáticos que produzem uma análise sintática correta com o algoritmo mencionado, de SSG (de "Simple Syntax Graph").

O compilador disponível para a linguagem PL/M-80 /INT 78/ só roda sobre o sistema operacional ISIS. Estamos orientando um projeto de implementação de um compilador dessa linguagem para o sistema operacional CPM, e decidimos utilizar nosso algoritmo de análise sintática. Para isso, foi necessário desenvolver um grafo SSG para essa linguagem, o que foi feito a partir da gramática BNF estendida apresentada em /INT 78/. Esse desenvolvimento mostrou ser um exercício não-trivial, tendo-se chegado a mudar a linguagem, expandindo-a em alguns pontos. Neste trabalho, apresentamos uma descrição intuitiva das restrições formais de /S-H 81/. Em seguida, apresentamos uma metodologia de desenvolvimento do grafo SSG para a linguagem PL/M-80, para finalmente apresentarmos o grafo propriamente dito.

2. A classe de grafos SSG

Esses grafos são conjuntos de subgrafos dirigidos

disconexos. Cada subgrafo corresponde a um não-terminal da gramática, representando as produções que definem esse não-terminal. Os nós do grafo são de dois tipos: nó terminal, representando por (t) ou (t) e nó não-terminal, representado por [N] onde t e N são os terminal e não-terminal contidos no nó, respectivamente. Por um abuso de linguagem, identificaremos algumas vezes o nó contendo o símbolo x como sendo o "nó x".

Cada nó tem uma só alternativa (ou um só nó alternativo), representada por uma aresta que é desenhada para baixo, saindo da aresta que chega em n. Assim, na fig. 1 M é o nó alternativo do nó a.

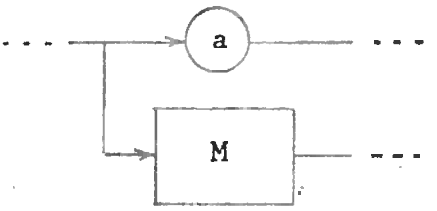


fig.1

Cada nó x tem no máximo um só sucessor (ou um só nó sucessor) apontado por uma aresta que sai de x. Na fig.2 M é o nó sucessor do nó a. Os fins de produção (em termos de gramática) são indicados por arestas apontando para um traço vertical, e são denominados fim de grafo. Assim, na fig.2 temos o subgrafo que representa a produção N→aM; note-se que concatenações são representadas por sequências de nós sucessores

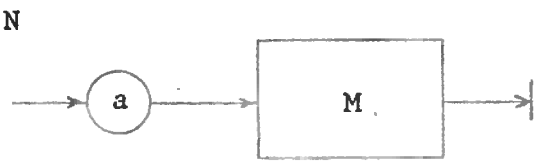


fig.2

Na fig.3, b é nó sucessor de nó a, e M sucessor de c. Os nós b e M não têm sucessor. Esse subgrafo corresponde às produções N→a(b|cM).

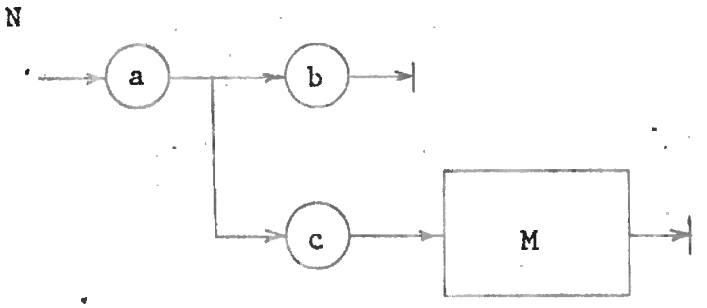


fig.3

Uma alternativa de um nó x que, em lugar de apontar para um nó aponta para um "fim de grafo" é denominada de alternativa vazia, como na fig.4. Em termos de geração, ela corresponde à geração de uma cadeia vazia.

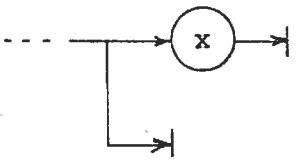


fig.4

Na fig.5, o nó b é nó alternativo e sucessor do nó a.

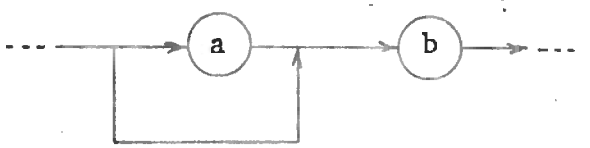


fig.5

Fechamentos de concatenação (repetição consecutiva de uma cadeia) são representados de duas maneiras. Na fig.6, apresentamos o fechamento $\{x\}^*$, onde x é um trecho qualquer de grafo. Na fig.7, apresentamos o grafo do fechamento $\{x \| y\}^*$ conforme notação introduzida em /S-H 81/, indicando a ocorrência de x pelo menos uma vez, e sua eventual repetição que deve ser precedida sempre de y, resultando a cadeia xyxy...yx com zero ou mais yx.

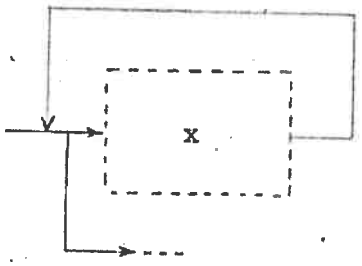


fig.6

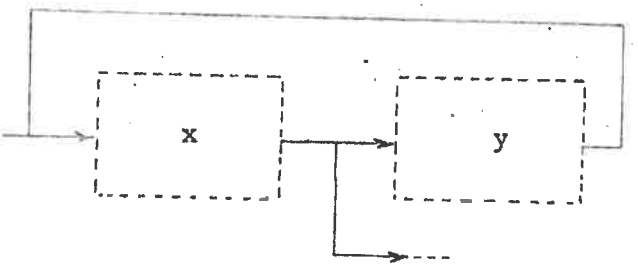


fig.7

Na fig.8, apresentamos um grafo que gera a linguagem (em termos de gramática) $S \overset{*}{\rightarrow} \{ ab^* \{ e^n f^n \mid n \geq 0 \} \| e \}^* d$.

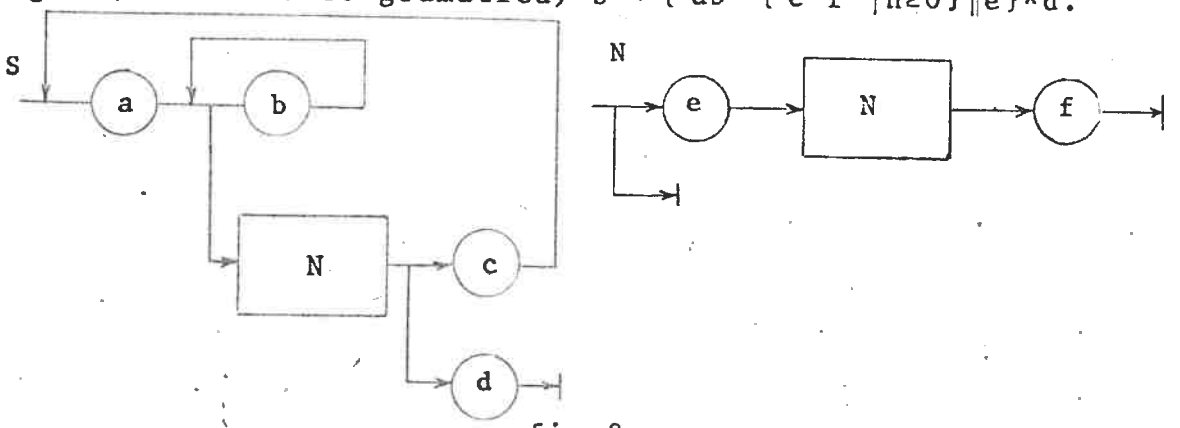


fig.8

Os grafos sintáticos representam as transições de um autômato a pilha que, interpretando o grafo, reconhece a linguagem "gerada" pelo mesmo. O algoritmo de reconhecimento é informalmente o seguinte. Suponhamos que o próximo símbolo de entrada seja s e que o próximo nó a ser interpretado pelo autômato seja o nó x. Se x contiver um terminal t, verifica-se se t=s, caso em que o símbolo seguinte a s é lido, passando a ser o próximo símbolo de entrada, e o próximo nó a ser interpretado deve ser o sucessor de x; nesse caso dizemos que o nó x e o símbolo s foram reconhecidos. Se t≠s, o próximo nó a ser interpretado deve ser a alternativa de x, iniciando-se novamente o processo descrito. Se x contiver um não-terminal N, deve-se iniciar recursivamente o processo iniciando-se pelo primeiro nó do subgrafo de N. Volta-se para o sucessor de x ao atingir-se (recursivamente) um fim de grafo. A recursão é eliminada empilhando-se em uma pilha um apontador para o nó x antes de desviar para o subgrafo de N; esse apontador permite a volta para o nó x, devendo-se então

desempilhar o seu apontador e passar para o sucessor de x. Nesse caso dizemos que N foi reconhecido.

Os grafos sintáticos que produzem uma análise sintática correta para qualquer cadeia gerada pelos mesmos, usando-se esse algoritmo, serão denominados de SSG (de "simple syntax graph"). Nem todos os grafos sintáticos são dessa classe. A razão reside no fato de que o algoritmo que interpreta o grafo para fazer uma análise sintática deve ser determinístico (para evitar "backtracking"). Esse não seria o caso de um trecho de grafo como o da fig.9. O algoritmo nunca testaria a última alternativa se a entrada fosse 'a'.

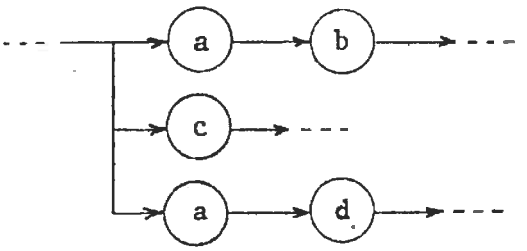


fig.9

Intuitivamente, a classe dos SSGs é composta dos grafos com as seguintes restrições.

a) Numa sequência de nós alternativos contendo exclusivamente terminais, todos esses terminais devem diferir entre si. A fig.9 é um contra-exemplo dessa regra.

b) Numa sequência de nós alternativos, nenhum nó da sequência pode ser alternativo de um nó da sequência. A fig.10 mostra dois desses casos proibidos.

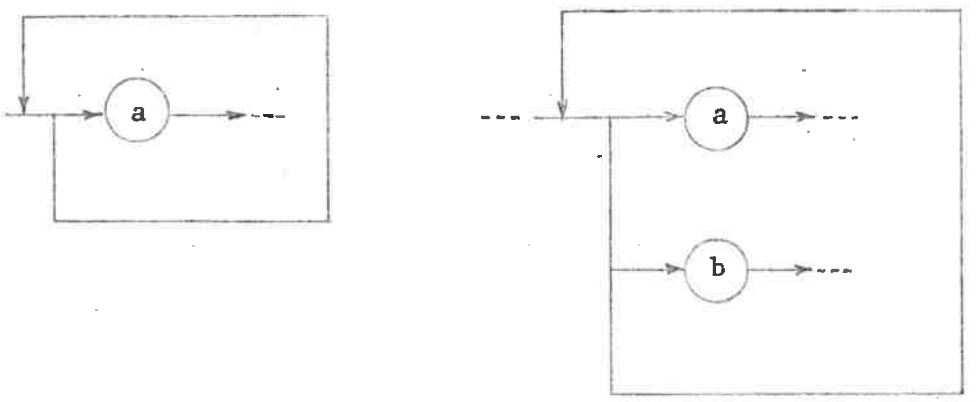


fig.10

c) Nenhum nó não-terminal pode ter nó alternativo ou alternativa vazia. A fig.11 mostra 3 casos proibidos

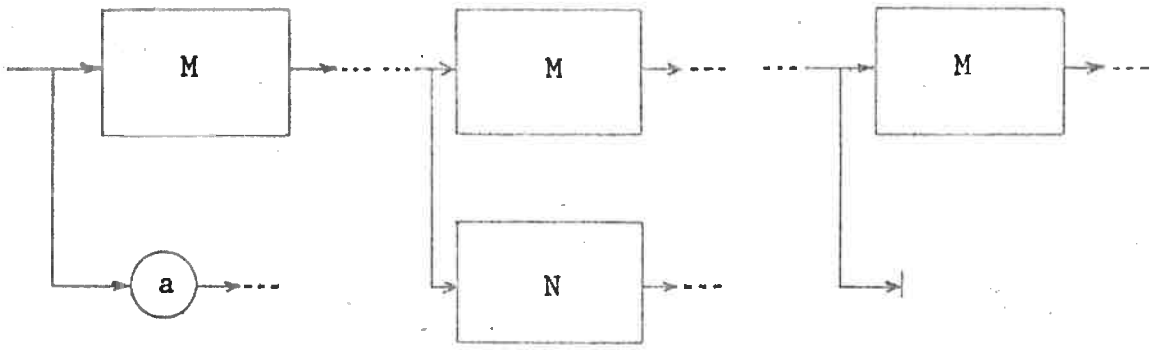


fig.11

d) Se um nó não-terminal contendo o símbolo N encerrar uma sequência de alternativas, cada terminal pertencente a essa sequência não pode ser gerado mais à esquerda por N. A fig. 12 mostra um caso proibido, pois M gera mais à esquerda os símbolos a e b (isto é, $M \overset{*}{\rightarrow} a...$, $M \overset{*}{\rightarrow} b...$) e a é o conteúdo de um dos nós da sequência de alternativas que termina em N.

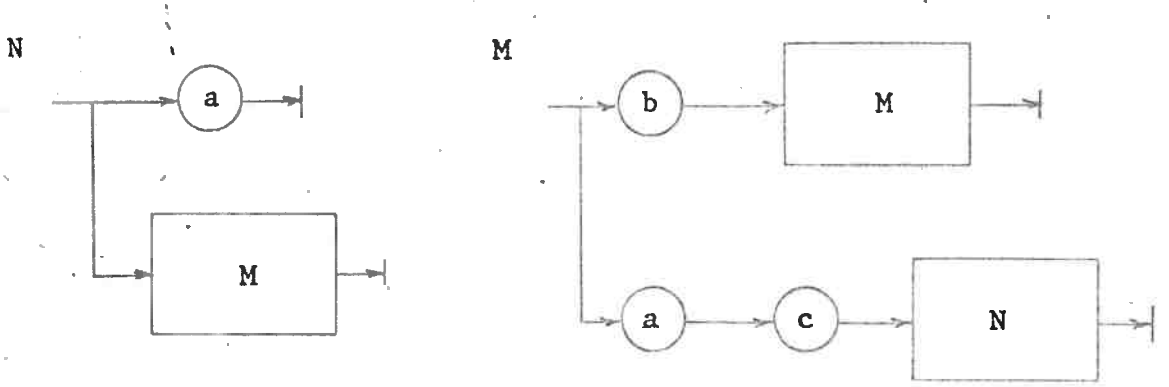


fig.12

e) Nenhum nó pode ser alternativa de uma alternativa vazia, como na fig.13.

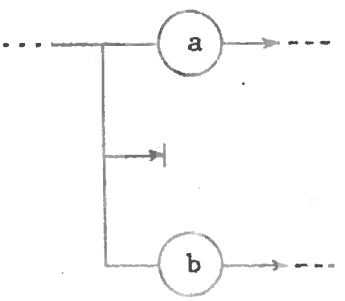


fig.13

f) Se o subgrafo de um não-terminal N tiver uma sequência de alternativas encerrada (i) por uma alternativa vazia ou (ii) por um nó contendo um não-terminal M tal que M gera uma cadeia vazia (isto é, $M \overset{*}{\rightarrow} \lambda$), então o conjunto de terminais dessa sequência de alternativas deve ser disjunto do conjunto dos terminais que podem seguir imediatamente N em qualquer forma sentencial (isto é, o conjunto $\{ t \mid S \overset{*}{\rightarrow} \ldots Nt \ldots \}$). A fig. 14 apresenta um exemplo de um caso (i) proibido; note-se que a sentença aa seria analisada erradamente pelo algoritmo (o símbolo inicial é S)

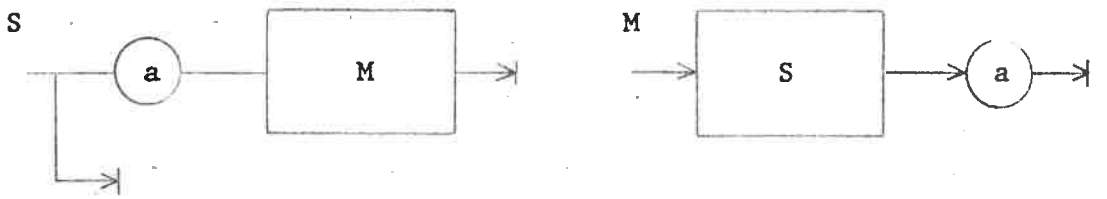


fig.14

g) Cada não-terminal deve gerar mais à esquerda pelo menos um símbolo terminal. O grafo da fig.15 apresenta um contra-exemplo a essa regra (supomos que não haja nenhum outro subgrafo nesse grafo).

fig.15

h) Um nó x contendo uma cadeia vazia só pode ocorrer como último nó de uma sequência de alternativas; se esse nó tem o nó y como sucessor, a cadeia de alternativas que é obtida eliminando-se x da sequência (isto é, tomando-se y como alternativa do nó que tem x como alternativa) deve satisfazer todos os itens de restrições, inclusive este (isto é, de (a) a (h)).

Note-se que as restrições (d), (f) e (g) impedem a recursividade (direta ou indireta) à esquerda, como nos casos das figs.15 e 16, respectivamente.

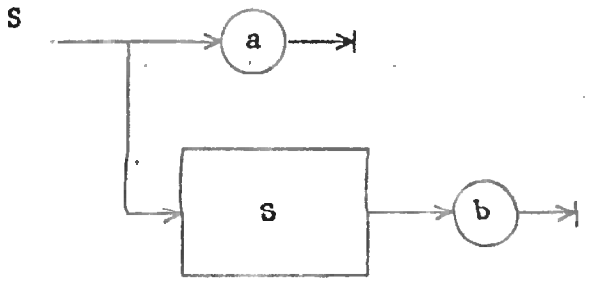


fig.16

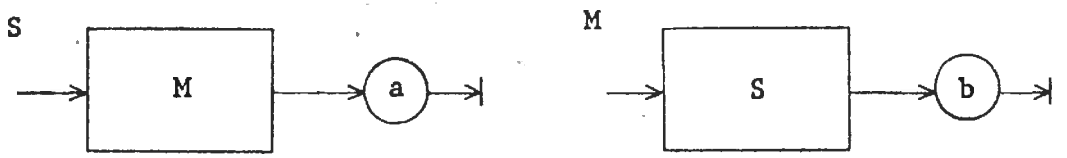


fig.17

Com estas restrições, obtemos grafos que correspondem a uma classe de gramáticas LL(1)/R-S 70/ estendidas, isto é, com expressões regulares nos lados direitos. Intuitivamente, ve-se que se trata de LL(1) pois em cada ponto de qualquer subgrafo o próximo símbolo de entrada determina univocamente o caminho a seguir. Se um não-terminal é alternativa de um nó, as restrições garantem que o ramo do primeiro sempre deve ser tomado, se for atingido.

É interessante notar que, em certos casos, pode-se ter um grafo correspondente a uma gramática ambígua; ele não satisfará as restrições dadas, mas haverá um reconhecimento "correto" seguindo sempre uma das possíveis análises. Este é o caso do exemplo da fig.18, cujo grafo não satisfaz a restrição f, para a cadeia aa.

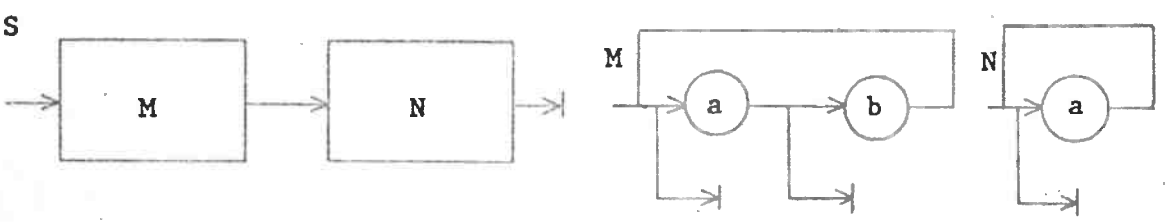


fig.18

3. Obtenção de SSG's a partir de gramáticas livres de contexto.

No projeto mencionado no item 1 enfrentamos o problema de derivar um grafo do tipo SSG para a linguagem PL/M-80. Para isso, partimos da gramática BNF estendida apresentada no Apêndice A de /INT 78/, tentando conservar nomes de não-terminais para facilitar o uso do manual. A tarefa mostrou-se não trivial, o que nos levou a desenvolver uma metodologia para essa derivação: trata-se de fazer substituições sistemáticas de nós não-terminais pelos seus sub-grafos e transformações equivalentes destes (isto é, sem alterar a linguagem gerada), até obterem-se as restrições apresentadas no item anterior. Infelizmente temos que exemplificar essa metodologia com um exemplo bem simples, para restringirmos seu tamanho.

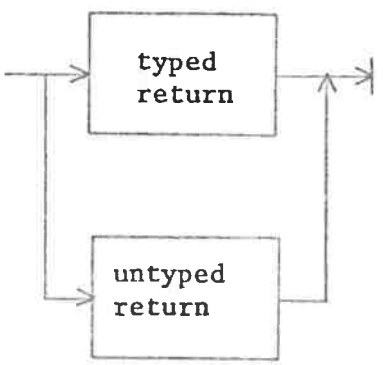
Seja a definição de "return statement", que é o item A.4.1.5 da BNF citada:

```
<return statement> ::= <typed return> |<untyped return>
<typed return> ::= RETURN <expression>;
<untyped return> ::= RETURN;
```

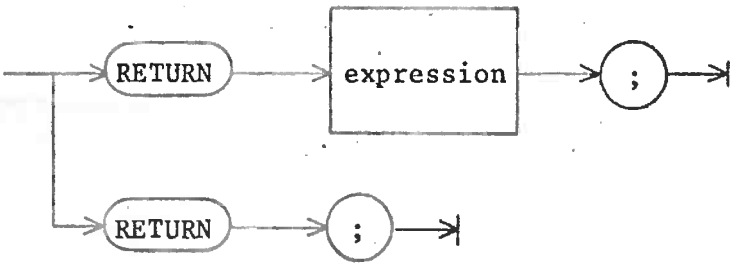
Os passos de derivação são dados na fig.19; note-se que no grafo apresentado no próximo item esse trecho está imerso em "unit"

return statement

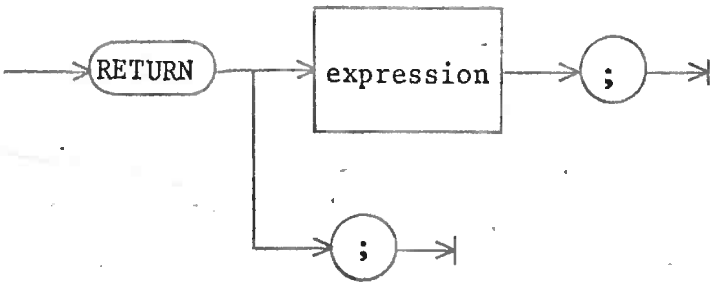
a)



b)



c)



d)

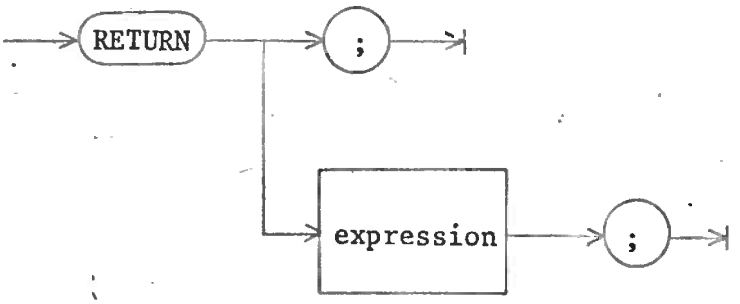


fig.19

4. Um grafo sintático para a linguagem PL/M-80

Neste item apresentamos o grafo sintático objeto do presente trabalho. As seguintes observações são pertinentes:

a) Foram introduzidos os seguintes não-terminais que não ocorrem na gramática de /INT 78/: "declaration sequence", "restricted minus", "unit sequence" e "reference part".

b) Foram introduzidas classes de identificadores em todo uso de identificadores que não sejam rótulos, em contraposição à sua declaração, onde eles aparecem com o terminal IDEN.

As classes são as seguintes:

- VAIDEN (de "variable identifier"); identificador declarado em um "declare element" (IDEN no subgrafo deste não-terminal);
- MEIDEN (de "member identifier"); identificador declarado em "variable type" dentro de um "structure";

- PRIDEN (de "procedure identifier"); declarado como identificador de procedimento em "procedure definition" sem tipo, isto é, sem retorno de valor tipo ADDRESS ou BYTE;
- FUIDEN (de "function identifier"): idem a PRIDEN, com retorno de valor.

Essas classes devem substituir os identificadores reconhecidos pelo Analisador Léxico antes de serem usados pelo Analisador Sintático, por meio de consulta à tabela de símbolos do compilador /S-H 81/.

c) Houve necessidade de alterar-se a sintaxe da "embedded assignement", pois da maneira como esta foi declarada em /INT 78/ a linguagem PL/M-80 não é de classe LL(k). Para isso "aumentamos" a precedência dessa atribuição, passando-a ao nível de "primary". Com isso, obtivemos uma extensão sintática da linguagem gerada pelo grafo, tornando-a um superconjunto da PL/M-80. De fato, agora é permitido escrever

A=B+C:=D+E

que é reconhecido como

A=B+(C:=D+E)

que por sinal também é permitido, mas que era a notação obrigatória na linguagem original. Evidentemente pode-se voltar à forma original por meio de rotinas "semânticas" do compilador, se for desejado.

d) Foi alterada a definição de "procedure definition", englobando algumas restrições dos atributos conforme o cap.8 de /INT 78/.

e) Foi alterada a definição de "restricted expression", tendo-se ajustado a mesma aos testes feitos com o compilador PL/M-80 disponível; note-se que a gramática original é ambígua nesse ponto.

f) A sintaxe expressa pelo grafo é ambígua. Uma sequência como

do; L:A=B; end

poderia ter o trecho 'L:' gerado tanto em "declaration sequence" como em "unit sequence". Note-se que o algoritmo de

análise sintática descrito no item 2 reconhecerá sempre o primeiro caso. Essa ambiguidade poderia ter sido evitada com perda da simplicidade do grafo.

g) A gramática original é ambígua também em relação à geração do "menos unário"; essa ambiguidade foi conservada no grafo.

Para maior facilidade no uso do grafo, elaboramos listas de referências cruzadas dos não-terminais e dos terminais, apresentadas a seguir.

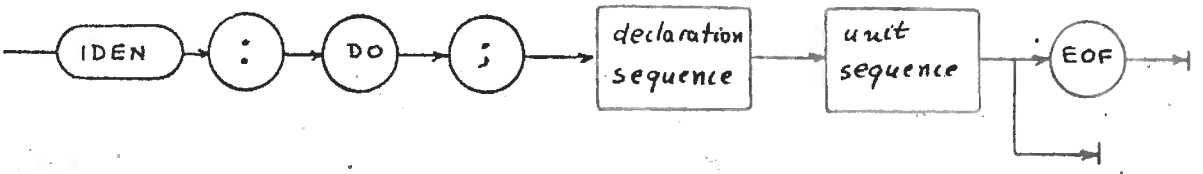
Referências cruzadas de não-terminais

Não-terminal	definição	uso (nº de vezes)
compilation	1	
declaration sequence	2	1, 11, 13
declare element	3	2
variable type	4	3(2)
variable attributes	5	3
initialization	6	3
base specifier	7	3(2)
restricted expression	8	5,6
restricted minus	9	8,10
restricted sum	10	8(2)
procedure definition	11	2
unit sequence	12	1, 11, 13
unit	13	12, 13
expression	14	13(8),20(2),19(3)
logical factor	15	14
logical primary	16	15
arithmetic expression	17	16
term	18	17
primary	19	18
reference part	20	13, 19(2)

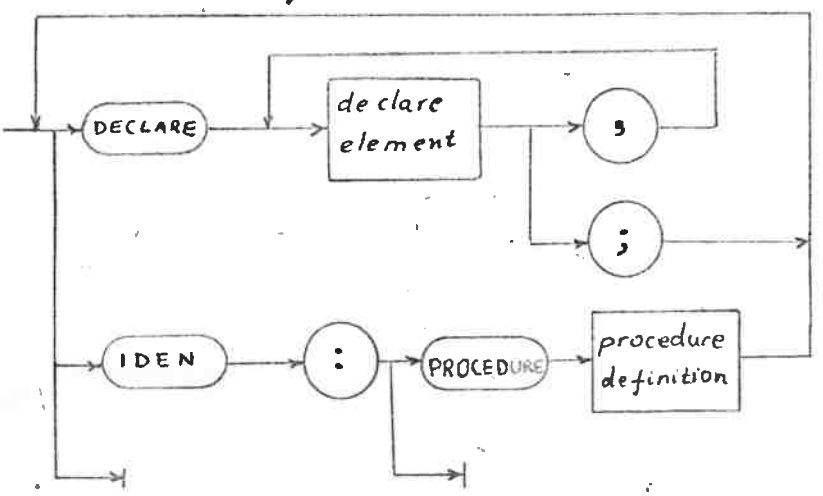
referências de terminais

terminal	subgrafo	terminal	subgrafo
ADDRESS	4(2), 11	RETURN	13
AND	15	STRING	3, 6, 19(2)
AT	5	STRUCTURE	4
BASED	3(2)	THEN	13
BY	13	TO	13(2)
BYTE	4(2), 11	VAIDEN	7, 8, 13(3), 19(2)
CALL	13	WHILE	13
CASE	13	:	1, 2, 12
DATA	6	;	1, 2, 11, 12, 13(4)
DECLARE	2	,	2, 3, 4, 6, 11, 13(2), 19(2)
DISABLE	13	(e)	3(4), 4(2), 5, 6, 8(2), 11, 13, 19(3), 20(2)
DO	1, 13		
ENABLE	13	*	3,18
ELSE	13	.	7, 8(2), 13(2), 19, 20
END	12	-	8, 9, 10, 17, 19
EXTERNAL	3(2), 5, 11	+	8, 10, 17
FUIDEN	12, 19(2)	=	13(2), 16
GO	13	<	16
GOTO	13	>	16
HALT	13	<=	16
IDEN	1,2, 3(2), 4, 11, 12		
IF	13	>=	16
INITIAL	6	<>	16
INTERRUPT	11	/	18
LABEL	3(2)	:=	19
LITERALLY	3		
MEIDEN	7,8, 13(2), 20		
MINUS	18		
MOD	19		
NOT	16		
NUMBER	3(3), 4, 10, 11, 19(2)		
OR	14		
PLUS	17		
PRIDEN	12, 13, 19		
PROCEDURE	2		
PUBLIC	3(2), 5, 11		
REENTRANT	11		

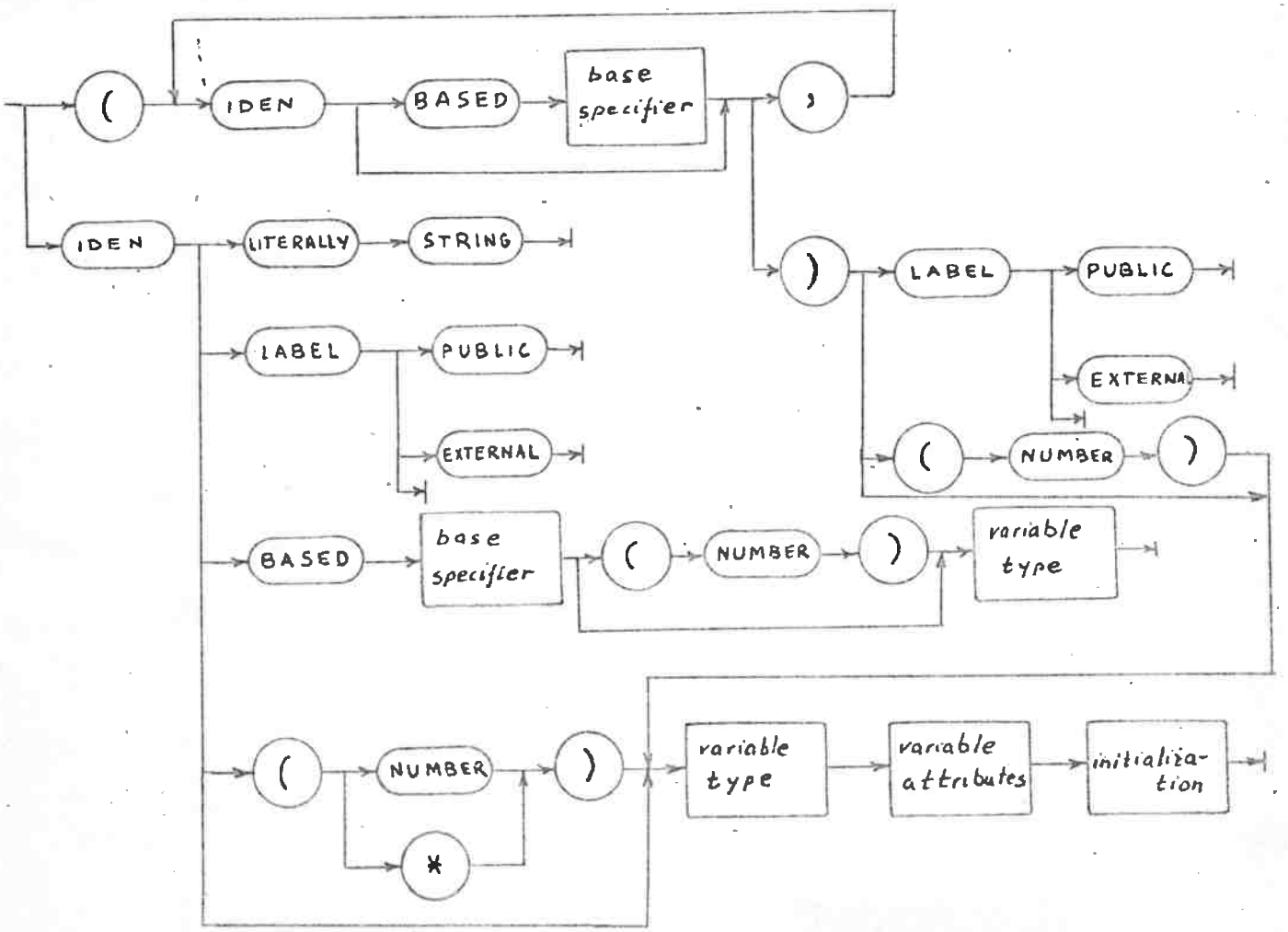
1. compilation



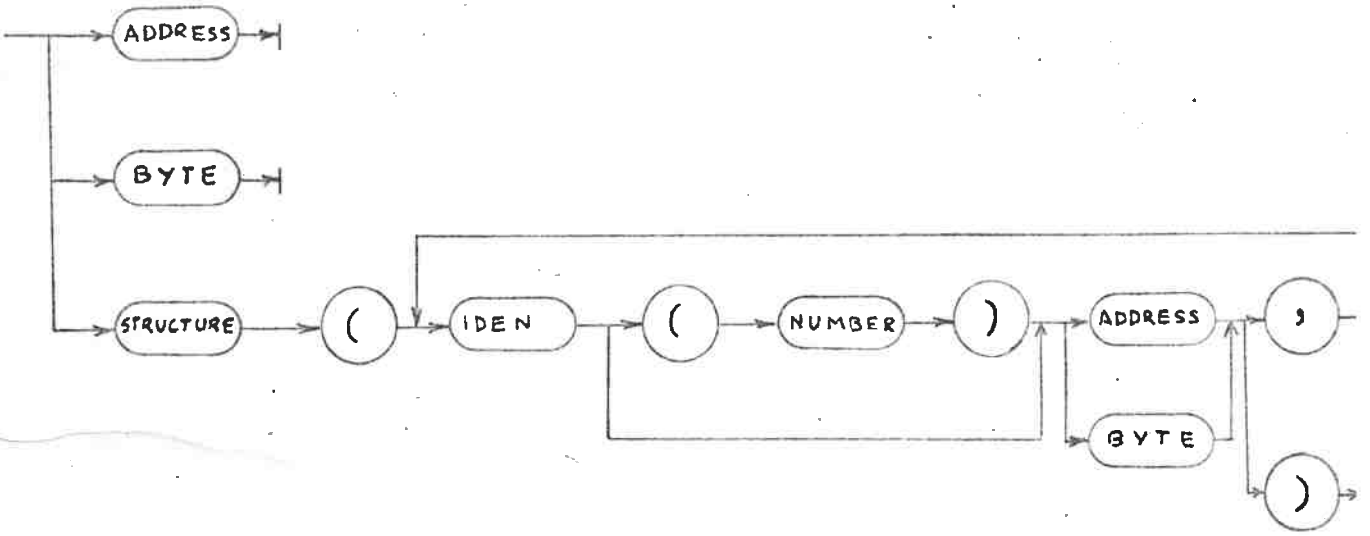
2. declaration sequence



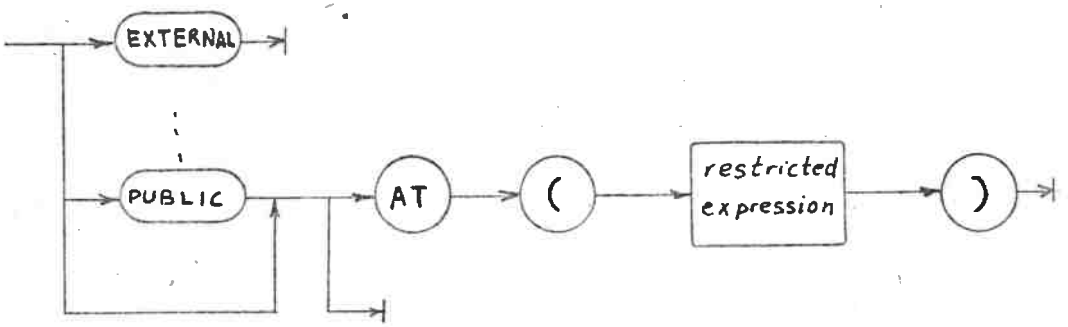
3. declare element



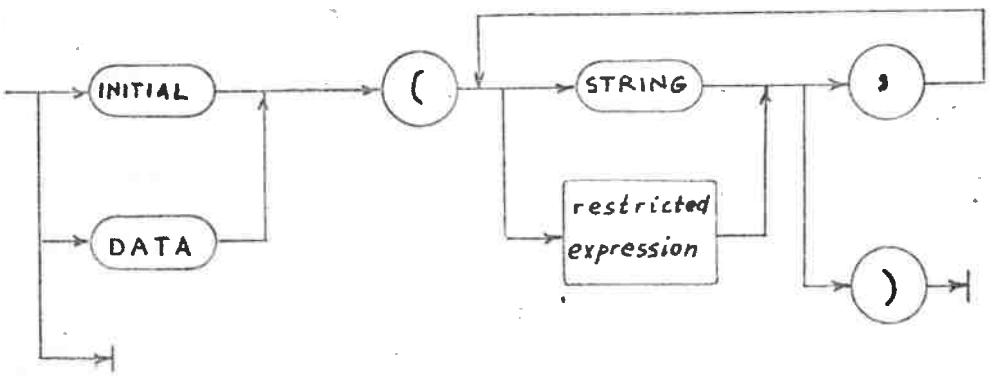
4. variable type



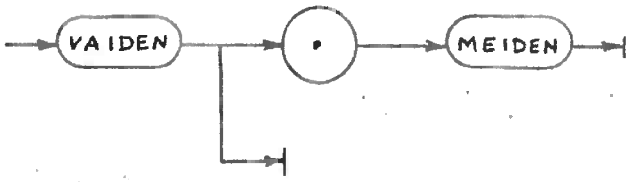
5. variable attributes



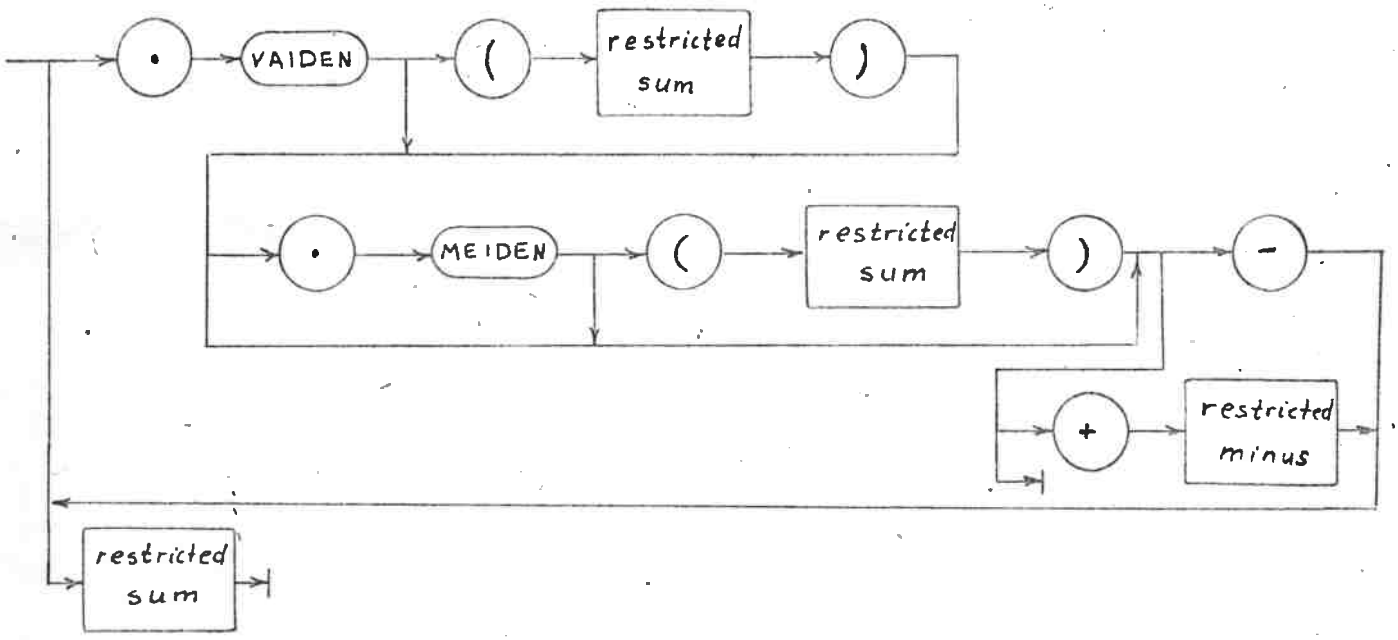
6. initialization



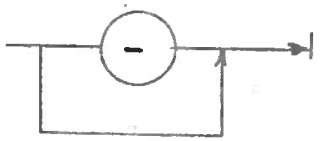
7. base specifier



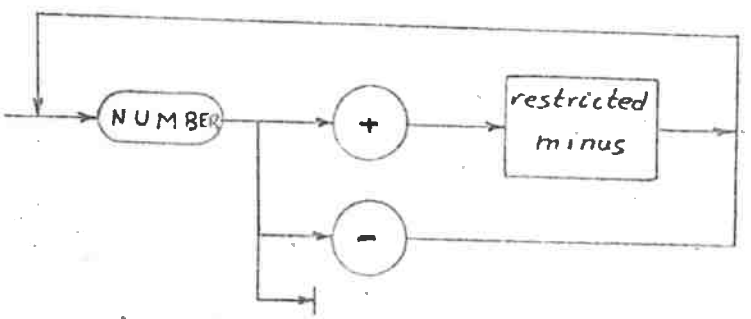
8. restricted expression



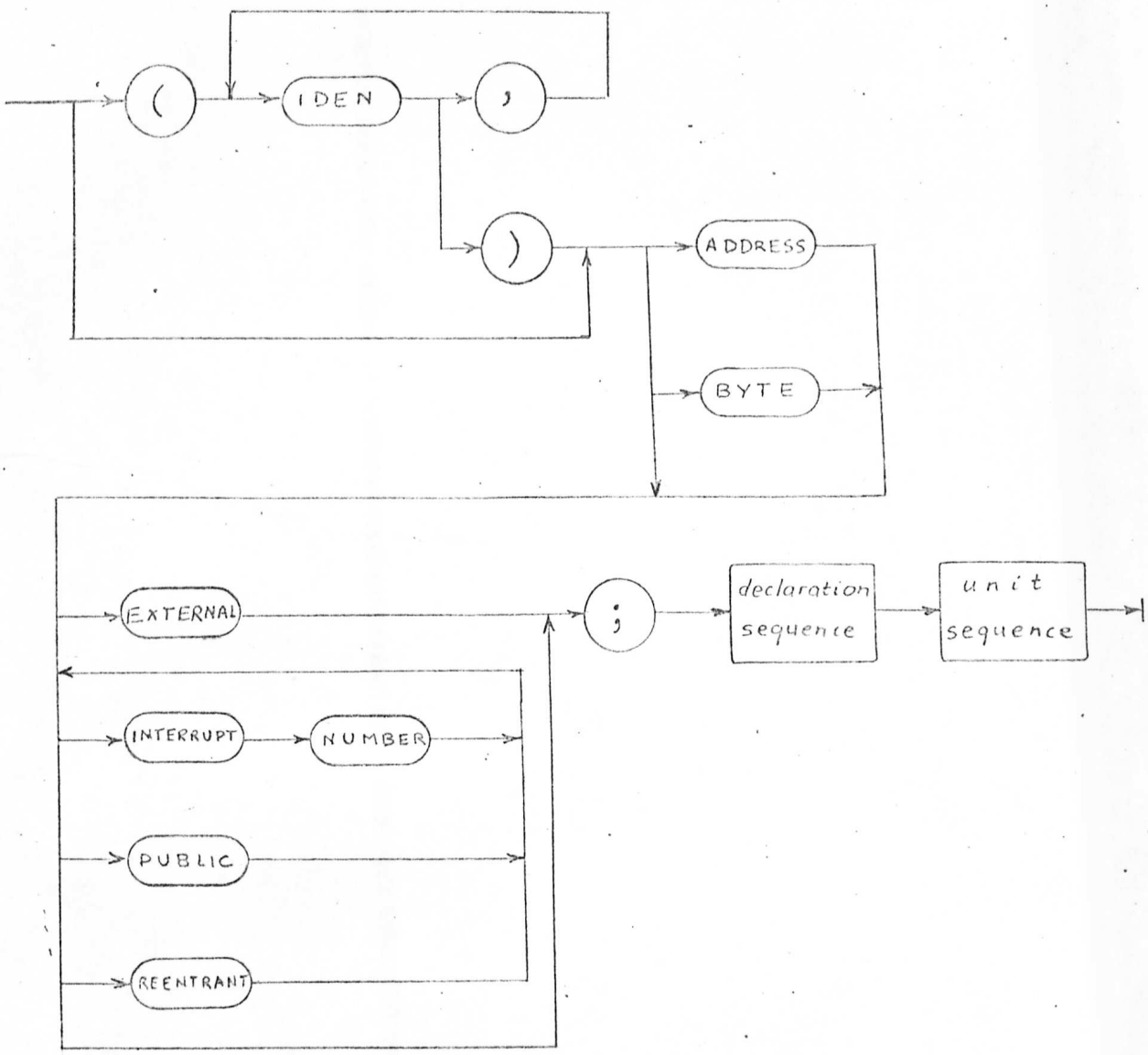
9. restricted minus



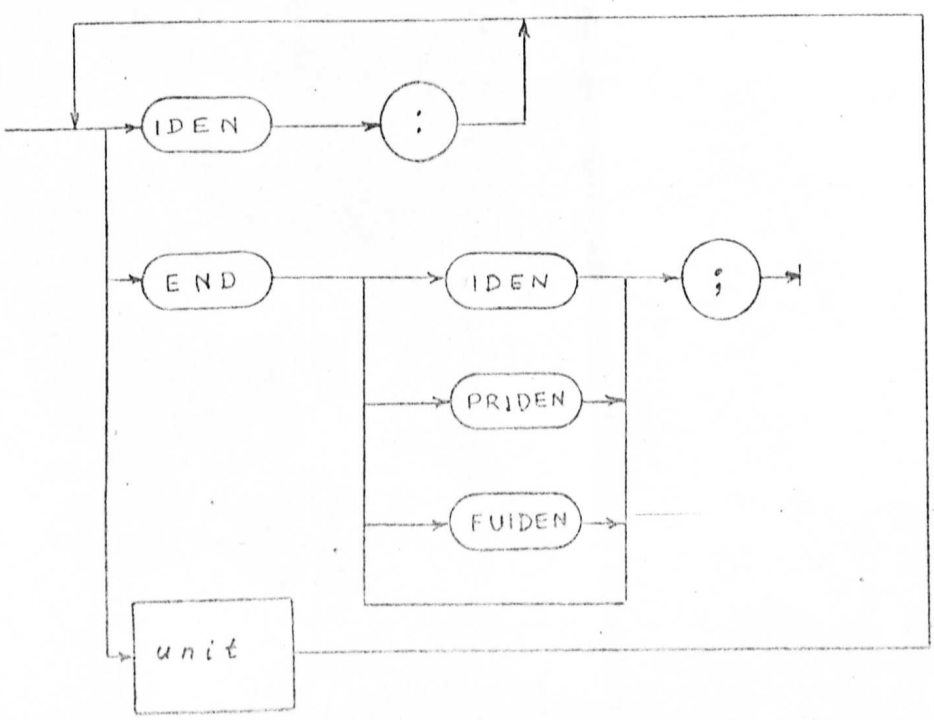
10. restricted sum



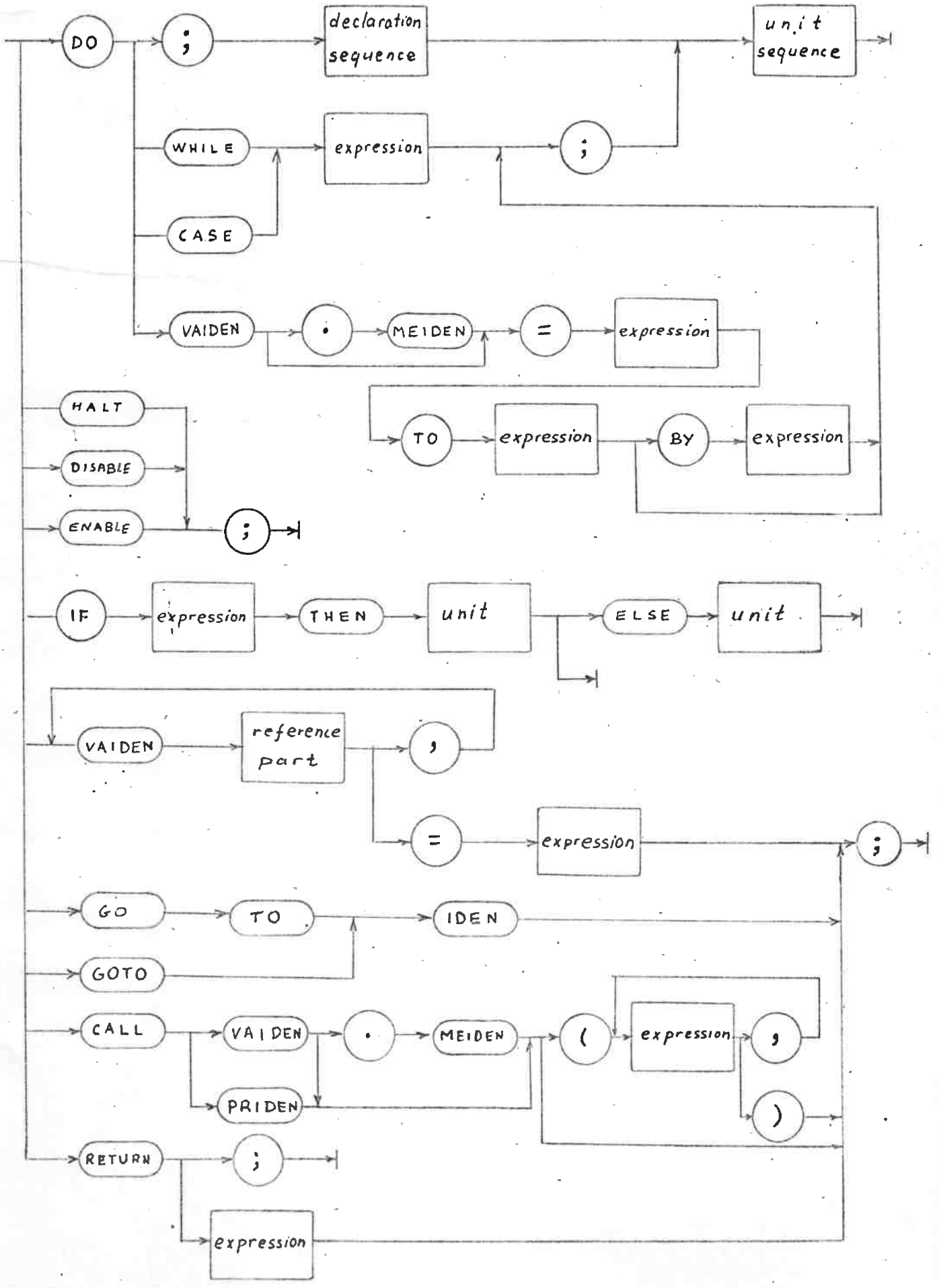
11. procedure definition



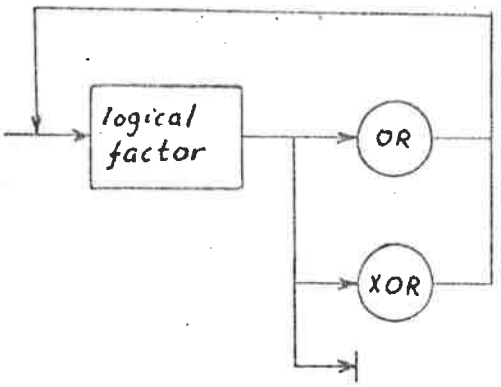
12. unit sequence



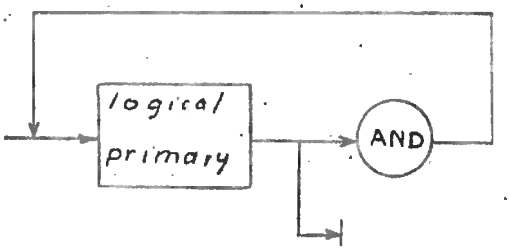
13. unit


DO
;
declaration sequence
unit sequence
WHILE
expression
;
CASE
VAIDEN
.
MEIDEN
=
expression
HALT
DISABLE
ENABLE
;
TO
expression
BY
expression
IF
expression
THEN
unit
ELSE
unit
VAIDEN
reference part
,
=
expression
;
GO
TO
IDEN
GOTO
CALL
VAIDEN
.
MEIDEN
(
expression
,
)
PRIDEN
RETURN
;
expression

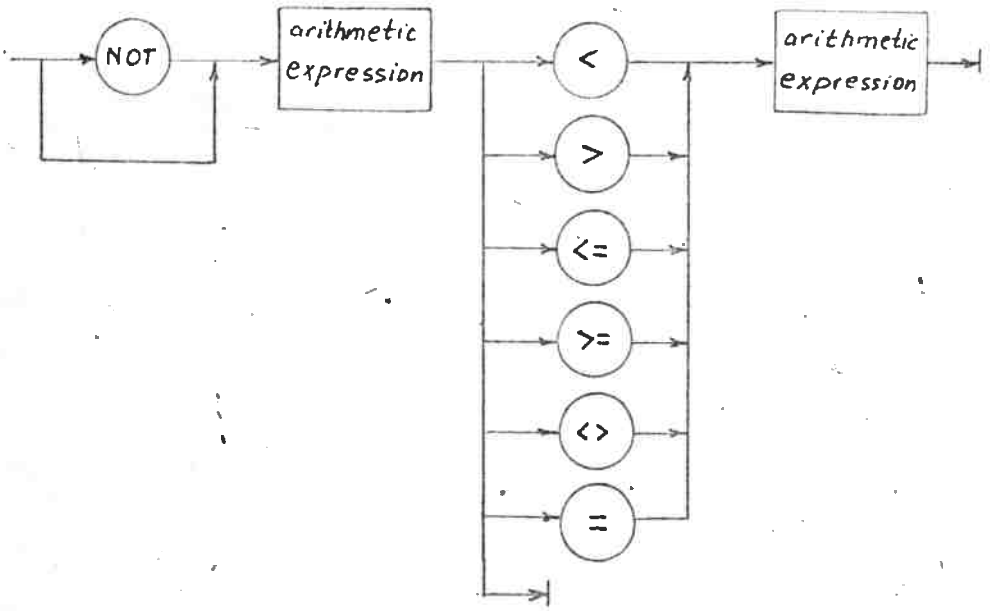
14. expression



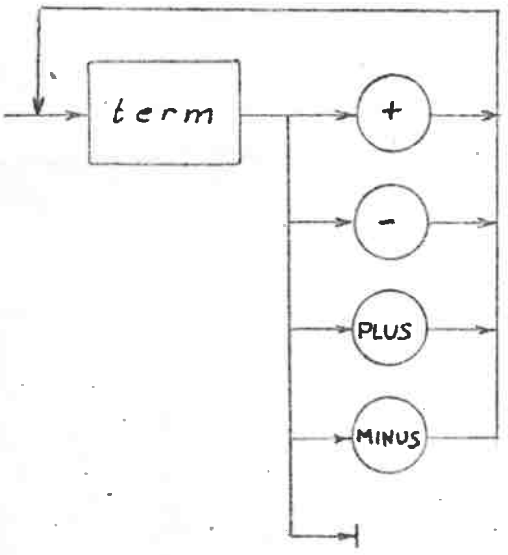
15. logical factor



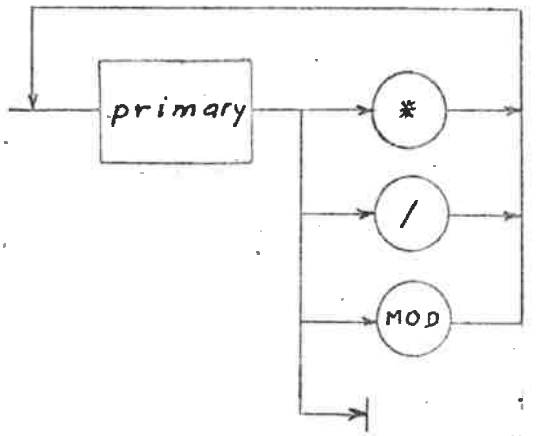
16. logical primary



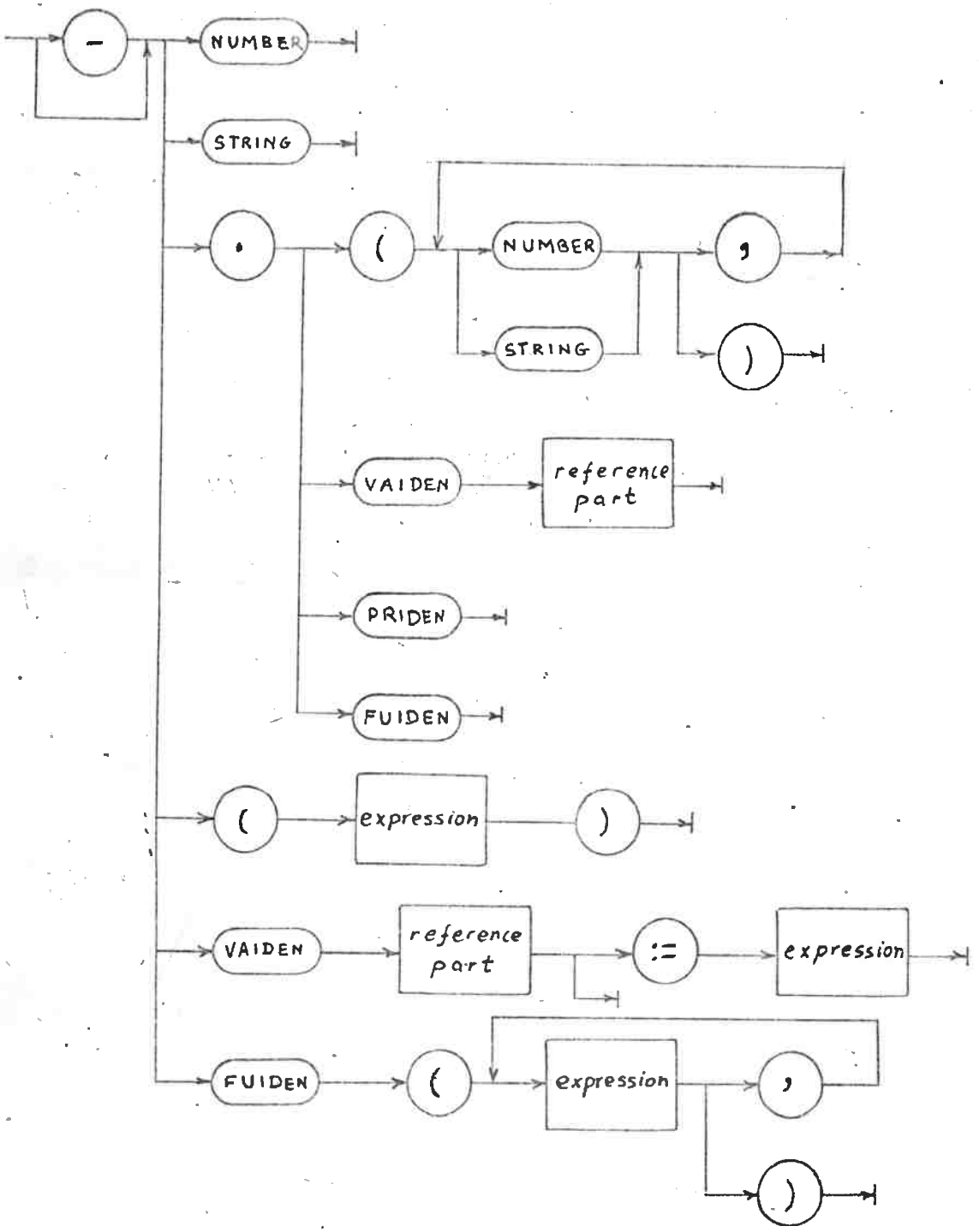
17. arithmetic expression



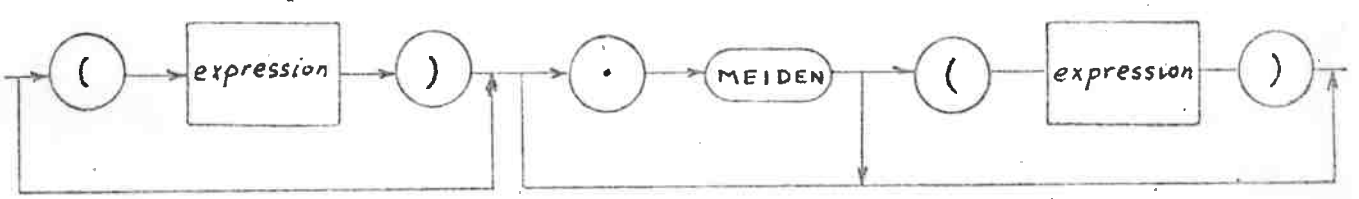
18. term



19. primary



20. reference part



Referências

/CON 63/ Conway, M.E. - Design of a separable transition diagram compiler, *Comm. ACM* Vol.4, 1961, pp. 51-55.

/GRI 71/ Gries, D. - *Compiler Construction for Digital Computers*, Wiley, New York 1971.

/INT 78/ *PL/M-80 Programming Manual*, Intel Corporation, Santa Clara, 1978.

/J-W 74/ Jensen, K. e Wirth, N. - *PASCAL user manual and report*, Springer Verlag, New York 1974.

/KOW 79/ Kowaltowski, I. - *Implementação de linguagens de programação*, Escola de Computação, IME-USP, São Paulo 1979.

/SET 79/ Setzer, V.W. - Non-recursive Top-down Syntax Analysis, *Software-Practice and Experience* Vol.9, 1979, pp.237-245.

/S-H 81/ Setzer, V.W. e Homem de Melo, I.S. - *A construção de um compilador*, 2ª Escola de Computação, Campinas, 1981 (2 Vols.) (no prelo, Ed. Campus, Rio de a Janeiro).

/WIR 71/ Wirth, N. - Design of a PASCAL compiler, *Software-Practice and Experience* Vol.1, 1971, pp-309-333.

TÍTULOS PUBLICADOS

RT-MAP-7811 - D.T. Fernandes e C. Patarra
Sistemas Lineares Esparsos, um Método Exato de Solução - Novembro 1978

RT-MAP-7812 - V.W. Setzer e G. Bressan
Desenvolvimento de Programas por Transformações: uma Comparação entre dois Métodos - Novembro 1978

RT-MAP-7813 - Ivan de Queiroz Barros
Variação do Passo na Discretização de Eq. Dif. Ord. com Condições Iniciais - Novembro 1978

RT-MAP-7814 - Martin Grötschel e Yoshiko Wakabayashi
On the Complexity of the Monotone Asymetric Travelling Salesman Polytope I: HIPOHAMILTONIAN FACETS - Dezembro 1978

RT-MAP-7815 - Ana F. Humes e E.I. Jury
Stability of Multidimensional Discrete Systems: State-Space Representation Approach - Dezembro 1978

RT-MAP-7901 - Martin Grötschel, Yoshiko Wakabayashi
On the complexity of the Monotone Asymetric Travelling Salesman Polytope II: HYPOTRACEABLE FACETS - Fevereiro 1979

RT-MAP-7902 - M.M. Sanches e V.W. Setzer
A portabilidade do Compilador para a Linguagem LEAL - Junho 1979

RT-MAP-7903 - Martin Grötschel, Carsten Thomassen, Yoshiko Wakabayashi
Hypotraceable Digraphs - Julho 1979

RT-MAP-7904 - N'Guiffo B. Boyom
Translations non triviales dans les groupes (transitifs) des transformations affines - Novembro 1979

RT-MAP-8001 - Ângelo Barone Netto
Extremos detectáveis por jatos - Junho 1980

RT-MAP-8002 - Ivan de Queiroz Barros
Medida e Integração
Cap. I - Medida e Integração Abstrata - Julho 1980

RT-MAP-8003 - Routo Terada
Fast Algorithms for NP-Hard Problems which are Optimal or Near-Optimal with Probability one - Setembro 1980

RT-MAP-8004 - V.W. Setzer e R. Lapyda
Uma Metodologia de Projeto de Bancos de Dados para o Sistema ADABAS
Setembro 1980

RT-MAP-8005 - Imre Simon
On Brzozowski's Problem: $(1uA)^m = A^*$ - Outubro 1980

RT-MAP-8006 - Ivan de Queiroz Barros
Medida e Integração
Cap. II - Espaços Lp - Outubro 1980

"RELATÓRIO TÉCNICO"
DEPARTAMENTO DE MATEMÁTICA APLICADA
TÍTULOS PUBLICADOS

RT-MAP-7701 - Ivan de Queiroz Barros
On equivalence and reducibility of Generating Matrices of RK-Procedures - Agosto 1977

RT-MAP-7702 - V.W. Setzer
A Note on a Recursive Top-Down Analizer of N.Wirth - Dezembro 1977

RT-MAP-7703 - Ivan de Queiroz Barros
Introdução a Aproximação Ótima - Dezembro 1977

RT-MAP-7704 - V.W. Setzer, M.M. Sanches
A linguagem "LEAL" para Ensino básico de Computação - Dezembro 1977

RT-MAP-7801 - Ivan de Queiroz Barros
Proof of two Lemmas of interest in connection with discretization of Ordinary Differential Equations - Janeiro 1978

RT-MAP-7802 - Silvio Ursic, Cyro Patarra
Exact solution of Systems of Linear Equations with Iterative Methods
Fevereiro 1978

RT-MAP-7803 - Martin Grötschel, Yoshiko Wakabayashi
Hypohamiltonian Digraphs - Março 1978

RT-MAP-7804 - Martin Grötschel, Yoshiko Wakabayashi
Hypotraceable Digraphs - Maio 1978

RT-MAP-7805 - W. Hesse, V.W. Setzer
The Line-Justifier: an example of program development by transformations
Junho 1978

RT-MAP-7806 - Ivan de Queiroz Barros
Discretização
Capítulo I - Tópicos Introdutórios
Capítulo II - Discretização
Julho 1978

RT-MAP-7807 - Ivan de Queiroz Barros
(Γ', Γ) - Estabilidade e Métodos Preditores-Corretores - Setembro 1978

RT-MAP-7808 - Ivan de Queiroz Barros
Discretização
Capítulo III - Métodos de passo progressivo para Eq. Dif. Ord. com condições iniciais - Setembro 1978

RT-MAP-7809 - V.W. Setzer
Program development by transformations applied to relational Data-Base queries - Novembro 1978

RT-MAP-7810 - Nguiffo B. Boyom, Paulo Boulos
Homogeneity of Cartan-Killing spheres and singularities of vector fields - Novembro 1978

TÍTULOS PUBLICADOS

RT-MAP-8101 - Luzia Kazuko Yoshida e Gabriel Richard Bitran
Um algoritmo para Problemas de Programação Vetorial com Variáveis Zero-Um - Fevereiro 1981

RT-MAP-8102 - Ivan de Queiroz Barros
Medida e Integração
Cap. III - Medidas em Espaços Topológicos - Março 1981

RT-MAP-8103 - V.W. Setzer, R. Lapyda
Design of Data Models for the ADABAS System using the Entity-Relationship Approach - Abril 1981

RT-MAP-8104 - Ivan de Queiroz Barros
Medida e Integração
Cap. IV - Medida e Integração Vetoriais - Abril 1981

RT-MAP-8105 - U.S.R. Murty
Projective Geometries and Their Truncations - Maio 1981

RT-MAP-8106 - V.W. Setzer, R. Lapyda
Projeto de Bancos de Dados, Usando Modelos Conceituais
Este relatório Técnico complementa o RT-MAP-8103. Ambos substituem o RT-MAP-8004 ampliando os conceitos ali expostos. - Junho 1981

RT-MAP-8107 - Maria Angela Gurgel, Yoshiko Wakabayashi
Embedding of Trees - August 1981

RT-MAP-8108 - Ivan de Queiroz Barros
Mecânica Analítica Clássica - Outubro 1981

RT-MAP-8109 - Ivan de Queiroz Barros
Equações Integrais de Fredholm no Espaço das Funções A-Uniformemente Contínuas - Novembro 1981

RT-MAP-8110 - Ivan de Queiroz Barros
Dois Teoremas sobre Equações Integrais de Fredholm - Novembro 1981

RT-MAP-8201 - Siang Wun Song
On a High-Performance VLSI Solution to Database Problems - Janeiro 1982

RT-MAP-8202 - Maria Angela Gurgel, Yoshiko Wakabayashi
A Result on Hamilton-Connected Graphs - Junho 1982

RT-MAP-8203 - Jörg Blatter, Larry Schumaker
The Set of Continuous Selections of a Metric Projection in C(X) - Outubro 1981

RT-MAP-8204 - Jörg Blatter, Larry Schumaker
Continuous Selections and Maximal Alternators for Spline Approximation - Dezembro 1981

RT-MAP-8205 - Arnaldo Mandel
Topology of Oriented Matroids - Junho 1982

RT-MAP-8206 - Erich J. Neuhold
Database Management Systems; A General Introduction - Novembro 1982

RT-MAP-8207 - Béla Bollobás
The Evolution of Random Graphs - Novembro 1982

TÍTULOS PUBLICADOS

RT-MAP-8208 - V.W. Setzer
Um Grafo Sintático para a Linguagem PL/M-80 - Novembro 1982