W-GRAMMAR

Guy de Chastellier
PROJET DE TRADUCTION AUTOMATIQUE
Alain Colmerauer
DEPARTEMENT D'INFORMATIQUE
UNIVERSITE DE MONTREAL, MONTREAL, CANADA

Summary

A new type of grammars is presented here, called W-grammars. It is shown how they can be used in translation processes. Examples are taken from the fields of algebraic manipulation and computational linguistics.

Introduction

In the Algol 68 report[1], A. van Wijngaarden used a new formalism to define the syntax of this language. We have called this formalism "W-grammar". It can be show that grammars of this type are very powerful since they can define every recursively enumerable set[2]. Moreover they can be used to describe complex string manipulations, and they seem to have good descriptive qualities. This is why, in a first approach, we have adopted them for our English-French machine translation project.

The experience we have had during the past year with W-grammars puts us in a favorable position to explain what they are and how they can be used in practical applications.

Definition of a W-grammar [1]

We presume the reader is familiar with the notions of string and vocabulary. As is usual, we denote by V^* the set of all strings over a vocabulary V. V^* contains the empty string, which is denoted by λ.

Any finite sequence of strings separated by commas will be called a complex string. If F denotes a set of strings then F^+ denotes the set of all the complex strings which can be constructed over F. F^+ contains the empty complex string, which is denoted by Λ.

A W-grammar is defined by:

(1) a finite vocabulary V_i the elements of which are called variables;

(2) a finite vocabulary V_u the elements of which are called values and such that $V_i \cap V_u = \phi$;

(3) a finite subset B of V_u^* the elements of which are called basic strings;

(4) a element of $V_i \cup V_u$ called axiom and usually denoted by S;

(5) a finite binary relation [2] on $(V_i \cup V_u)^*$ denoted by $\Rightarrow$ and such that

$$x \Rightarrow y \quad \text{implies} \quad x \in V_i;$$

the elements of $\Rightarrow$ are called meta-rules;

(6) a finite binary relation on $((V_i \cup V_u)^*)^+$ denoted by $\vdash\!\!\Rightarrow$ and such that

$$r \vdash\!\!\Rightarrow s \quad \text{implies} \quad r \in (V_i \cup V_u)^* \quad \text{and} \quad s \neq \Lambda;$$

the elements of $\vdash\!\!\Rightarrow$ are called pseudo-rules.

The relation $\Rightarrow$ will be extended as follows [3]:

$$x \Rightarrow y \quad \text{implies} \quad vxw \Rightarrow vyw \quad \text{for any}$$

$$x;y;v;w \in (V_u \cup V_i)^*$$

From the reflexive transitive closure [4] $\overset{*}{\Rightarrow}$ of the extended relation $\Rightarrow$, and from $\vdash\!\!\Rightarrow$, we can now define a new binary relation $\Longrightarrow$ on $(V_u^*)^+$:

$r \Longrightarrow s \equiv$ There exist $r';s' \in ((V_i \cup V_u)^*)^+$ such that $r' \vdash\!\!\Rightarrow s'$ and such that r and s can be respectively obtained from r' and s' by substituting for each occurrence of any variable U a string $t \in V_u^*$ such that $U \overset{*}{\Rightarrow} t$. If U occurs more than once in r' and/or s', the same string t has to be substituted at each occurrence.

[1] This is our own definition of a W-grammar; it differs from van Wijngaarden's, but it should describe the same system.

[2] We define a binary relation on a set E as a subset ρ of the Cartesian product E x E. For convenience $a \rho b$ is written instead of $(a, b) \in \rho$.

[3] If x and y denote strings, then xy denotes the string obtained by writing the string denoted by x on the left of the string denoted by y. Of course $x\lambda = \lambda x = x$ for any string x.

[4] The reflexive transitive closure ρ^* of a binary relation ρ is defined as:

$$a \rho^* b \equiv \left[\begin{array}{l} a = b \\ \text{or} \\ \text{there exists c such that } a \rho^* c \text{ and} \\ c \rho b \end{array}\right]$$

The elements of $\Rightarrow$ will be called rules. So the meta-rules permit the construction, from each pseudo-rule, of a possibly infinite number of rules.

In the same way as we have extended the relation $\longrightarrow$, we will extend the relation $\Rightarrow$[5] thus:

$r \Rightarrow s$ implies $p,r,q \Rightarrow p,s,q$ for any $r;s;p;q \in (V_u^*)^+$

The language L defined by a W-grammar is then:

$$L = \left\{ t \in B^+ \;\middle|\; \text{there exists } s \in V_u^* \text{ with } S \longrightarrow^* s \Rightarrow^* t \right\}$$

Any string $s \in V_u^*$ such that $S \longrightarrow^* s$ will be called an axiomatic string.

Example 1

The following W-grammar defines the well-known non-context-free language L, each element of which is of the form:

$\underbrace{a,a,\ldots,a}_{n \text{ times}}, \underbrace{b,b,\ldots,b}_{n \text{ times}}, \underbrace{c,c,\ldots,c}_{n \text{ times}}$ with $n>o$

variables: N;L.

values: a;b;c;u.

basic strings: a;b;c.

axiom: N.

meta-rules: N $\longrightarrow$ u
N $\longrightarrow$ Nu
L $\longrightarrow$ a
L $\longrightarrow$ b
L $\longrightarrow$ c

pseudo-rules: N $\mapsto$ Na,Nb,Nc
NuL $\mapsto$ NL,L
uL $\mapsto$ L

From the meta-rules and the pseudo-rules we can deduce the rules:

u $\Rightarrow$ ua,ub,uc
uu $\Rightarrow$ uua,uub,uuc
uuu $\Rightarrow$ uuua,uuub,uuuc
.....
uua $\Rightarrow$ ua,a
uub $\Rightarrow$ ub,b
uuua $\Rightarrow$ uua,a
uuub $\Rightarrow$ uub,b
.....
ua $\Rightarrow$ a
ub $\Rightarrow$ b

The axiomatic strings are:

u
uu
uuu
....

It is by considering each string as a symbol, and each complex string as a string of symbols, that we can conceive of the above rules as being "context-free"; and consequently we are able to draw the syntaxtic tree (fig. 1) corresponding to the derivation from uu into a,a,b,b,c,c.

It should be noted that the grammar not only defines the language L but also relates each element of the language to the axiomatic string(s) from which it is derived. The next example will illustrate this point.

Example 2 [6]

The following W-grammar describes all the arithmetical expressions which can be formed from the operands a,b,c and the operators +,-,x,/,↑,), (. The axiomatic strings are all the corresponding arithmetical expressions written in Polish suffix notation. The sign ⊖ is used to denote the unary operator. Variables are represented by symbolic names in capital letters, and values by symbolic names in small letters or by signs. If there are several successive rules with the same left-hand member, then the member is not repeated after the first of the rules.

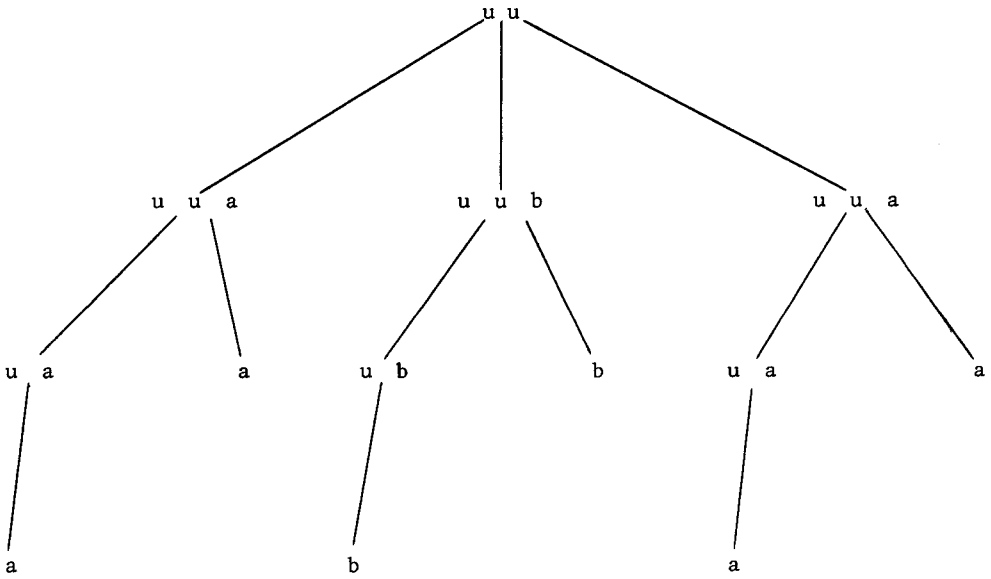
Basic strings: a;b;c;+;-;x;/;↑;);(.

Axiom: AXIOM

Meta-rules:
AXIOM $\longrightarrow$ S expression
S $\longrightarrow$ S ⊖
$\longrightarrow$ S S PLUS
$\longrightarrow$ S S TIMES
$\longrightarrow$ S S ↑
$\longrightarrow$ P
PLUS $\longrightarrow$ +
$\longrightarrow$ -
TIMES $\longrightarrow$ x
$\longrightarrow$ /
P $\longrightarrow$ a
$\longrightarrow$ b
$\longrightarrow$ c
T $\longrightarrow$ S

[5] If r and s denote complex strings, than r,s denotes the complex string obtained by writing the complex string denoted by r, followed by a comma, on the left of the complex string denoted by s. Of course $r,\Lambda = \Lambda,r = \Lambda$ for any string r.

[6] A similar example is given in [3] to illustrate transduction grammars. There seems to be a connection between W-grammars and transduction grammars.


u u
u u a
u u b
u u a
u a
a
u b
b
u a
a
a
b
a

Pseudo-rules:

```
          S  expression  ⟼> S  term
                         ⟼> +  ,  S  term
        S  ⊖ expression  ⟼> -  ,  S  term
S  T  PLUS  expression   ⟼>
          S  expression  ,  PLUS  ,  T  term
               S  term   ⟼> S  factor
     S  T  TIMES  term   ⟼>
          S  term  ,  TIMES  ,  T  factor
           S  factor     ⟼> S  primary
      S  T  ↑  factor    ⟼>
           S  factor  ,  ↑  ,  T  primary
          S  primary     ⟼>
                (  ,  S  expression  ,  )
          P  primary     ⟼> P
```

Figure 2 shows how the axiomatic string "ab + ab - x expression" is related to the arithmetic expression "(,a,+b,),x,(,a,-,b,)".

Example 3

Our aim in this example is to emphasize how use can be made of the fact that W-grammars define a mapping from axiomatic strings to complex strings. If we use Chomsky's terminology[4], each axiomatic string can be considered in some way representative of the deep structure of the corresponding completely derived complex strings, while the latter represent the surface structures. So we can say informally that W-grammars are capable of describing deep structures together with the transformations needed to rearrange them into surface structures.

Even though the following example does not claim to solve a linguistic problem, it is interesting for showing how two W-grammars, describing respectively a natural source language and a target one, can be linked in an automatic translation process. The reader will find much more sophisticated examples in [5] and [6].

We must first assume that there exists a level of deep structure at which the structures are common to both languages. The idea then is to define two W-grammars such that they have a common set of axiomatic strings and such that these strings describe the deep structures just referred to. These axiomatic strings serve as an 'intermediate language'. So to translate a sentence, it is a matter of first analyzing it with the W-grammar of the source language, then taking the axiomatic strings that are obtained and generating from them the translation in the target language by means of the other W-grammar. It follows that if the sentence to be translated has several interpretations according to the grammar, several axiomatic strings will be produced and several translations will result.

Below we give a W-grammar of English that describes sentences of the following four patterns and derives them from deep structures (i.e. axiomatic strings):

The boy gives a book to the girls
The boy gives the girls books
The book is given to a girl by the boy
The boy is given the books by a girl

Variations are possible in the choice of the nouns, of the definite or indefinite article, of the plural or the singular. Subsequently the W-grammar of French given further on synthesizes a French translation out of each axiomatic string. Figure 3 provides an example showing how these two grammars work when linked. It consists of a translation of the English sentence

"the boy is given books by a girl"

into the French sentence

"des livre s sont donn é s au garçon par une fille"

Grammar of English

```
Meta-rules: P   —> SN  SV
           SV   —> MODE  V  SN  à  SN
         MODE   —> actif
                —> passif
           SN   —> ART  NOMBRE  NOM
          ART   —> def
                —> indef
       NOMBRE   —> sing
                —> plu
          NOM   —> GENRE  N
        GENRE   —> mas
                —> fem
            N   —> garçon
                —> fille
                —> livre
            V   —> donn
          SN1   —> SN
          SN2   —> SN
        FORME   —> pp
                —> NOMBRE  actif
           CO   —> SN
                —> à  SN
```

Pseudo-rules:

inversion of the direct and the indirect objet

```
SN  MODE  V  SN1  à  SN2  ⟼>
                    SN  MODE  V  SN2  SN1
```

construction of the active form

```
SN1  actif  V  SN2  CO  ⟼>
          SN1  actif  V  ,  SN2  ,  CO
```

construction of the passive form

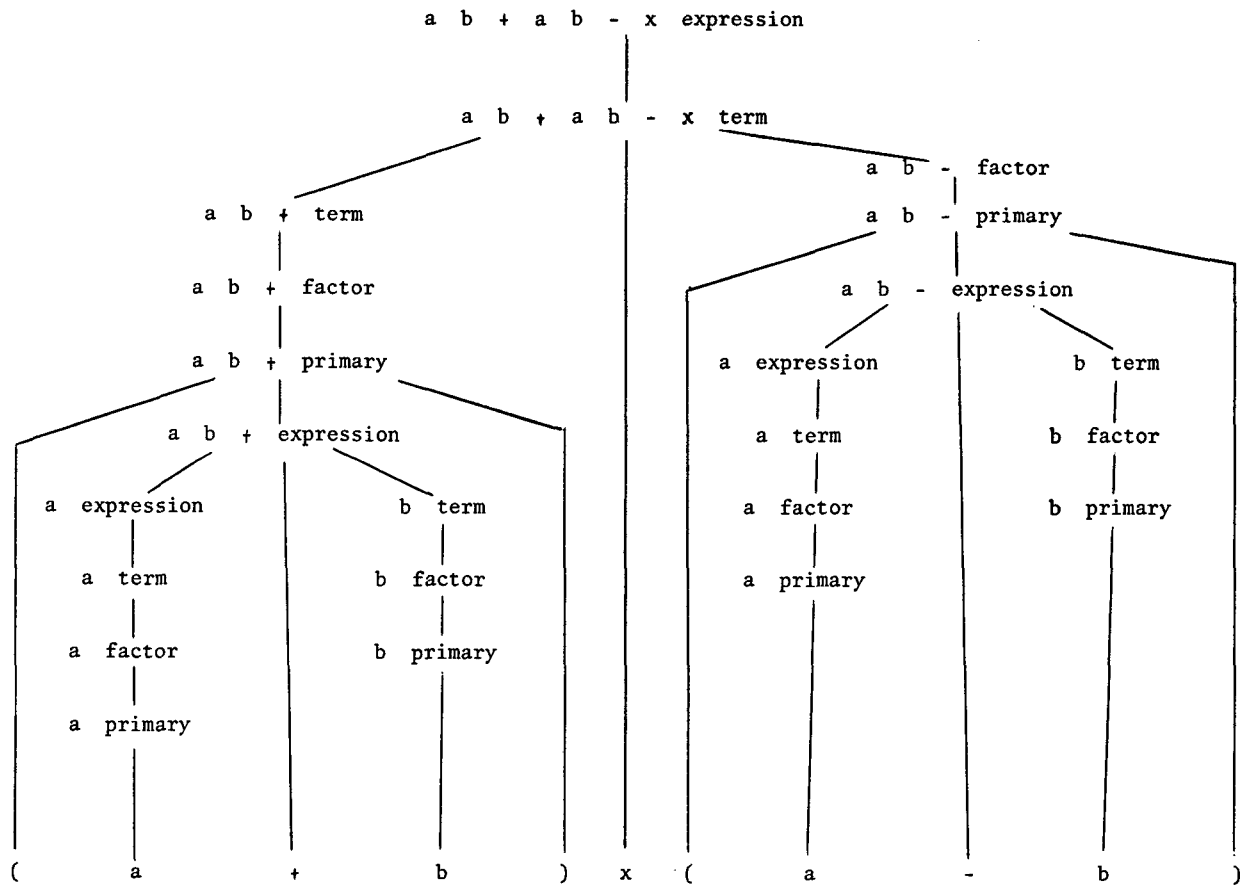
```
SN1  passif  V  SN2  CO  ⟼>
      SN2  passif  V  ,  CO  ,  by  ,  SN1
```

agreement of the subject with the verb

```
ART  NOMBRE  NOM  MODE  V  ⟼>
      ART  NOMBRE  NOM  ,  NOMBRE  MODE  V
```

construction of the indirect object

```
                à  SN  ⟼> to  ,  SN
```


a b + a b - x expression
a b + a b - x term
a b + term
a b + factor
a b + primary
a b + expression
a expression
a term
a factor
a primary
b term
b factor
b primary
a b - factor
a b - primary
a b - expression
a expression
a term
a factor
a primary
b term
b factor
b primary
(a + b) x (a - b)

verbal forms

```
NOMBRE passif V  ⟼ NOMBRE be , pp V
     FORME donn  ⟼ FORME give
sing actif give  ⟼ gives
 plu actif give  ⟼ give
        pp give  ⟼ given
        sing be  ⟼ is
         plu be  ⟼ are
```

forms of the article and its agreement with a noun

```
def NOMBRE NOM  ⟶ the , NOMBRE NOM
indef sing NOM  ⟶ a , sing NOM
 indef plu NOM  ⟶ plu NOM
```

forms of the noun, and a part of lexicon

```
NOMBRE mas livre  ⟼ NOMBRE book
NOMBRE mas garçon ⟼ NOMBRE boy
NOMBRE fem fille  ⟼ NOMBRE girl
       sing book  ⟼ book
        plu book  ⟼ books
        sing boy  ⟼ boy
         plu boy  ⟼ boys
       sing girl  ⟼ girl
        plu girl  ⟼ girls
```

Grammar of French

Meta-rules:

```
P    ⟶ SN SV
SV   ⟶ MODE V SN à SN
MODE ⟶ actif
     ⟶ passif
SN   ⟶ ART NOMBRE NOM
ART  ⟶ def
     ⟶ indef
NOMBRE ⟶ sing
       ⟶ plu
NOM  ⟶ GENRE N
GENRE ⟶ mas
      ⟶ fem
N    ⟶ garçon
     ⟶ fille
     ⟶ livre
V    ⟶ donn
SN1  ⟶ SN
SN2  ⟶ SN
BON-ART ⟶ def sing fem
        ⟶ indef NOMBRE GENRE
AART ⟶ ART
     ⟶ à ART
```

Pseudo-rules:

construction of the active form

```
SN actif V SN1 à SN2 ⟼
          SN actif V , SN1 , à SN2
```

construction of the passive form

```
SN passif V SN1 à SN2 ⟼
       SN1 passif V , à SN2 , par, SN
```

subject-verb agreement

```
ART NOMBRE GENRE N MODE V ⟼
 ART NOMBRE GENRE N , NOMBRE GENRE MODE V
```

construction of the article eventually preceded by the preposition à

```
AART NOMBRE GENRE N ⟼
      AART NOMBRE GENRE , NOMBRE N
       def sing mas  ⟶ le
       def sing fem  ⟶ la
      def plu GENRE  ⟶ les
à     def sing mas  ⟶ au
        à BON-ART   ⟼
                      à , BON-ART
      indef sing mas ⟶ un
      indef sing fem ⟶ une
     indef plu GENRE ⟶ des
à    def plu GENRE   ⟶ aux
```

construction of the verbal forms

```
NOMBRE GENRE passif V ⟼
     NOMBRE être , NOMBRE GENRE pp V
      plu GENRE pp V ⟼
                  sing GENRE pp V , s
       sing fem pp V ⟼
                    sing mas pp V , e
      sing mas pp V  ⟶ V é
sing GENRE actif V   ⟶ V e
 plu GENRE actif V   ⟶ V ent
          sing etre  ⟶ est
           plu etre  ⟶ sont
```

construction of the noun

```
sing N ⟼ N
 plu N ⟼ N s
```

Machine Implementation of W-grammars

We have written and tested programs [7] implementing W-grammars on a CDC 6400. Our system consists essentially of an analyzer and a synthesizer; both are written in Algol 60.

The input of the analyzer is a complex string t, and its output consists of all the axiomatic strings s such that

$$s \Rightarrow^{*} t.$$

Two restrictions are imposed:

a) if a variable occurs in the left-hand member of a pseudo-rule, it must also occur in the right-hand member;

b) there must be no infinite sequence of complex strings $r_1; r_2; \ldots$ such that

$$\ldots \Rightarrow r_2 \Rightarrow r_1 \Rightarrow t.$$

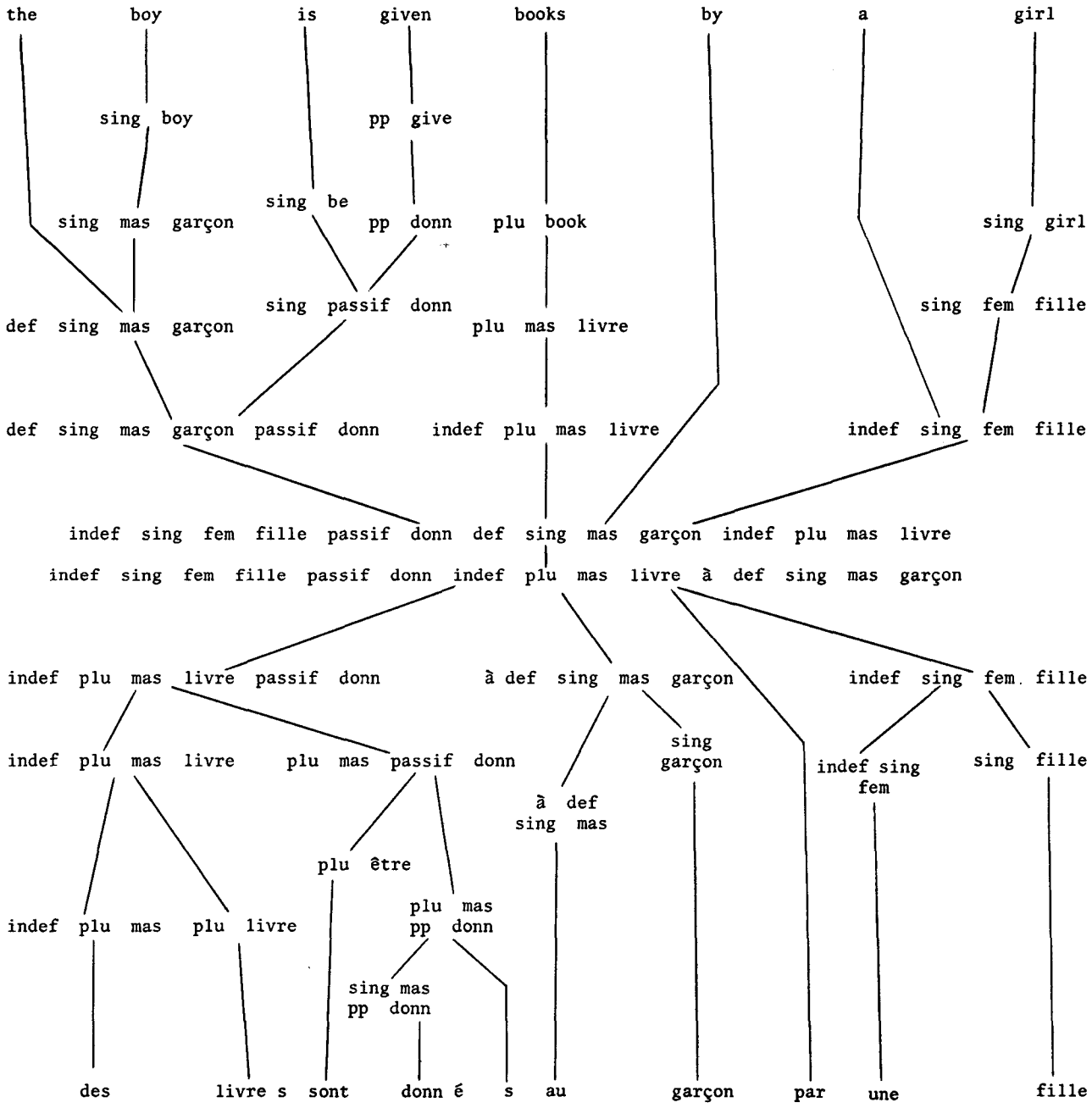
The input of the synthesizer is an axiomatic string s, and the output is the set of complex strings t such that

$$s \Rightarrow^{*} t.$$

Two restrictions are imposed:

a) if a variable occurs in the right-hand member of a pseudo-rule, it must also occur in the left-hand member;

[7] With the assistance of G. Stewart and C. Springer.


the boy is given books by a girl
sing boy pp give
sing be
sing mas garçon pp donn plu book sing girl
sing passif donn sing fem fille
def sing mas garçon plu mas livre
def sing mas garçon passif donn indef plu mas livre indef sing fem fille
indef sing fem fille passif donn def sing mas garçon indef plu mas livre
indef sing fem fille passif donn indef plu mas livre à def sing mas garçon
indef plu mas livre passif donn à def sing mas garçon indef sing fem fille
indef plu mas livre plu mas passif donn sing garçon indef sing fem sing fille
à def sing mas
plu être
indef plu mas plu livre plu mas pp donn
sing mas pp donn
des livre s sont donn é s au garçon par une fille

b) there must be no infinite sequence of complex strings $r_1;r_2;\ldots$ such that

$$t \Rightarrow r_1 \Rightarrow r_2 \Rightarrow \ldots$$

With the present system we are able to run grammars having a maximum of around 300 rules altogether, including meta-rules and pseudo-rules. The maximum acceptable length of the input strings into either the analyzer or the synthesizer is about 30 symbols. The time required for processing such a string is approximately 1 minute 30 seconds. Obviously the efficiency of the programs could be greatly increased by using disks and machine language. However the present system was only intended to be experimental, and so we have not cared to optimize it.[8]

References

1. VAN WIJNGAARDEN, A. (ed.), Mailloux, B.J., Peck, J.E.L., Koster, C.H.A., Final Draft Report on the Algorithmic Language Algol 68, Amsterdam: Mathematisch Centrum, December 1968.

2. SINTZOFF, M., Existence of a Van Wijngaarden syntax for every recursive enumerable set, Annales de la Société Scientifique de Bruxelles, Bruxelles, 81, II (1967), pp. 115-118.

3. LEWIS, P.M., and STEARN, R.F., Syntax directed transduction, JACM, vol. 15, no.3, July 1968.

4. CHOMSKY, N., Three models for the description of language, I.R.E. Transactions on Information Theory, vol. IT-2, Proceedings of the Symposium on Information Theory, Sept. 1956.

5. GOPNIK, M., Relative Clauses, Recherche sur la traduction automatique, onzième rapport semestriel, Université de Montréal et Conseil National de la recherche, Montréal, October 1968.

6. Recherche sur la traduction automatique, douzième rapport semestriel, Université de Montréal et Conseil National de la Recherche, Montréal 1969. (In preparation)

[8] The authors wish to thank B. Harris for his help in preparing the English version of this paper.