ma
the
ma
tisch

cen
trum

MR 123/71

WIJNGAARDEN A van

On the Boundary between Natural and Artificial Languages.

amsterdam 1971

Abstract of the paper "On the boundary between natural and artificial languages" by A. van Wijngaarden.

The method of syntactic description used for the new programming language ALGOL 68 has proved its power in that domain. It appears to have also a field of application in the analysis of sentences and their meaning in natural languages. Of course, the method may be used to give production rules for notions like *word* or *sentence* in order to display the syntactic structure of a word or a sentence. However, since one is not restricted to context-free constructions, a new and interesting field is opened by taking a whole sentence as a notion and by transforming that sentence into another one by successive application of production rules each transforming a sentence into another one with, by definition, the same meaning. If no production rule is anymore applicable, then the resulting sentence may be such that one can associate a meaning with it. This meaning is then also that of the original sentence and the syntax has served to display this meaning. In particular, it is shown how a sentence like *how much is 691 - 365?* can be transformed into its answer *326*. . Since the correct answer to a question requires the understanding of all meaning, if any, contained in the question, it seems that, at least in this borderline area between formal and natural languages, semantic analysis and syntactic analysis are two names for one same process.

On the boundary between natural and artificial languages

by

A. van Wijngaarden

The purpose of this paper is to show the application of a new method of syntactic description to a number of examples. The first examples are syntactic definitions as might occur in the definition of a programming language but they gradually become more like examples taken from a natural language, stylished English, and the last ones are complete questions in the English language, though admittedly belonging to a very restricted class. The method of description was proposed by the author in [1] and later extensively used to define the syntax of ALGOL 68 [2]. In experimenting with the tool, it became apparent that it may be of considerable value in the analysis of natural languages also. Actually, it was found that the method not only can describe parts of the syntax of a natural language, but also is able to describe the transformation of certain sentences into other ones which reveal their semantical contents.

Our first concept is that of a "protonotion". A protonotion is any sequence of "small syntactic marks". In [2] those small syntactic marks are small letters, but here, since we want to treat whole sentences in a natural language as protonotions, we shall accept as small syntactic marks also the other marks occurring in those sentences, e.g., digits, punctuation marks and operators, i.e., all recognizable symbols except the capital letters in some type font, for which we choose italics here. Examples of protonotions are then, e.g., *how much is*, *691* and *how much is 691 - 365?*. The blanks occurring in these protonotions are irrelevant and only inserted to help the reader to do the necessary parsing.

Our second concept is that of a notion, i.e., a protonotion for which a "production rule" is given. A production rule consists of that notion fol-

lowed by ":" followed by a "direct production" of that notion, i.e., a, possibly empty, sequence of members, i.e., protonotions, separated by "," and closed by ".", e.g.,

identifier : *letter*.

identifier : *identifier* , *letter*.

identifier : *identifier* , *digit*.

Production rules for one same notion may be abbreviated, using a ";"; e.g., the three production rules given above may be abbreviated into the rule

identifier : *letter* ; *identifier* , *letter* ; *identifier* , *digit*. .

Such a rule is also referred to as a production rule.

A production of a notion is either a direct production or is obtained by replacing one of the member protonotions in a production of that notion which member is itself a notion by a direct production of that member; e.g., *letter* , *letter* obtained from *identifier* , *letter* by replacing *identifier* by its direct production *letter*. A terminal production of a notion is a production of that notion in which none of the members is a notion. The reader may have some representation associated with each such member in a terminal production and then the sequence obtained by replacing each member of a terminal production by its representation and obliterating any separating commas is a representation of the terminal production. In the definition of programming languages, such a member may be *letter a symbol*, with which the reader may associate a mark on paper representing the letter "a" in some type font or a pattern of holes in a punched card representing the letter "a" in some code. In dealing with natural languages, such a member may be *horse*, with which the reader may associate the image of a particular animal. Of course, the reader may not have available a representation for one or more members and then he has only an incomplete representation of the termi-

nal production, which is a quite common situation in the interpretation of sentences in a natural language.

Our third concept is that of a metanotion, i.e., a nonempty sequence of "large syntactic marks". In accordance with [2], we choose these large syntactic marks to be capital letters, here in italics, since this is here, as well as in [2], sufficient and yields very readable rules. Also production rules for metanotions are provided. A production rule for a metanotion consists of that metanotion followed by ":" followed by a direct production of that metanotion and closed by ".". A direct production of a metanotion is either empty or a list of metamembers separated by blanks, a metamember being a metanotion or a protonotion. A production of a metanotion is either a direct production of that metanotion or is obtained by replacing in a production of that metanotion one of the metamembers which is a metanotion by one of its direct productions. A production of a metanotion is terminal if it is empty or if all its metamembers are protonotions. For instance, the production rules for the metanotion *AAA*

AAA : *a*.

AAA : *AAA a*. ,

also abbreviated to

AAA : *a* ; *AAA a*. ,

yield as terminal productions of the metanotion *AAA* the protonotions *a*, *aa*, *aaa*, etc.

At last, we introduce the concept of a hyperrule, i.e., a rule obtained by inserting in a production rule for a notion one or more metanotions. Such a hyperrule stands for any production rule which can be obtained by replacing each metanotion in the hyperrule by one of its terminal productions with the restriction that a metanotion which occurs more than once in the

hyperrule is replaced by the same terminal production at all those occurrences.

We now show the consequences of these definitions at the hand of a number of examples. As a first example, we define the notion *letter* by means of the hyperrule

letter : *letter LETTER symbol*. , (N1)

where the metanotion *LETTER* is defined by the production rule

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*. , (M1)

which stands for the 26 production rules obtained by replacing *LETTER* by one of the terminal productions given in (M1), yielding

letter : *letter a symbol*. ,

letter : *letter b symbol*. ,

and so on. In itself, rules (N1) and (M1) are only somewhat shorter than the 26 production rules obtained. However, the metanotion *LETTER* may also occur in other hyperrules and the saving is therefore much greater than it might seem to be at first sight. For example, we can define the notion *palindrome*, i.e., a word which reads the same from left to right as from right to left, as follows

palindrome : *letter* ;
letter LETTER symbol , *letter LETTER symbol* ;
letter LETTER symbol , *palindrome* , *letter LETTER symbol*. (N2)

Here we see an example of the importance of the restriction that a metanotion in a hyperrule should be replaced by one same terminal production consistently. Of course, the notion *palindrome* is still definable by a context free grammar, though at the cost of 53 production rules. Our next example, however, is essentially different. Let a *sandwich word* be a sequence of

letter symbols consisting of a certain number of times a certain letter symbol followed by the same number of times a second letter symbol followed again by the same number of times the first letter symbol, e.g., *aba*, *xxx*, or *ppppqqqqpppp*. First of all, we need to distinguish between two arbitrary, not necessarily different, letter symbols. To this end, we introduce a metanotion *LATTER* by means of the production rule

LATTER : *LETTER*. . (M2)

The terminal productions of *LATTER* are therefore the same as but independent of those of *LETTER*. Moreover, we must be able to recognize the length of a sequence and we do this in a crude way by means of the metanotion *LENGTH*, defined by the production rule

LENGTH : *one* ; *one plus LENGTH*. . (M3)

This metanotion *LENGTH* has therefore an infinite number of terminal productions, wiz., *one*, *one plus one*, *one plus one plus one*, etc. To someone who knows the English language these terminal productions strongly suggest the most primitive definition of the natural numbers, but we shall not make use of this fact. With this definition the rest is easy:

row of LETTERs of length one : *letter LETTER symbol*. (N3)

row of LETTERs of length one plus LENGTH :

letter LETTER symbol , *row of LETTERs of length LENGTH*. (N4)

sandwich word : *row of LETTERs of length LENGTH* ,

row of LATTERs of length LENGTH , *row of LETTERs of length LENGTH*. (N5)

Observe that (N4) and (N5) each stand for an infinite number of production rules because of the fact that *LENGTH* has an infinite number of terminal productions. This fact, together with the fact that we can enforce a same terminal production in different places of a production rule, enables us to produce notions like *sandwich word* which cannot be produced by a context

free grammar. In fact, Sintzoff showed in [3] that any recursively enumerable set may be generated by some such a syntax. Of course, it is not always obvious how to define an intuitively conceived notion. For instance, if we want to produce a *proper sandwich word* by which is meant a *sandwich word* whose middle sequence consists of other letter symbols than its first and last sequences, then the production is more difficult since we have a built-in mechanism to enforce equality of terminal productions but not one to enforce unequality. This problem is overcome by ranking the terminal productions to be distinguished in the following way:

mark one : letter a symbol. , (N6.1)

mark one plus one : letter b symbol. , (N6.2)

mark one plus one plus one : letter c symbol. , (N6.3)

and so on. Then we can write with the aid of the metanotions *RANK* and *SHIFT* defined by

RANK : LENGTH. (M4)

SHIFT : LENGTH. (M5)

the following hyperrules

row of marks RANK of length one : mark RANK. (N7)

row of marks RANK of length one plus LENGTH :

mark RANK , row of marks RANK of length LENGTH. (N8)

proper sandwich word :

row of marks RANK plus SHIFT of length LENGTH ,

row of marks RANK of length LENGTH ,

row of marks RANK plus SHIFT of length LENGTH ;

row of marks RANK of length LENGTH ,

row of marks RANK plus SHIFT of length LENGTH ;

row of marks RANK of length LENGTH. (N9)

Rule (N9) shows a new feature. By choosing for *RANK* and *SHIFT* sufficiently small numbers, viz., so that *RANK plus SHIFT* does not exceed, say, 26, a *proper sandwich word* is produced in terms of *letter a symbol*, *letter b symbol* up to *letter z symbol*. If one chooses *RANK plus SHIFT* higher, then one is faced with a protonotion *mark one plus one plus ∿ ∿ ∿ plus one*, for which no production rule is given, although the notion does not end with *symbol*. When defining a programming language, like ALGOL 68, such a situation is a blind alley. When studying the meaning of a sentence in a natural language, one has to take a different attitude. One may consider the sentence as written with small syntactic marks. This means that the sentence is a notion. This notion may produce, by means of certain production rules all having, by definition, the same meaning. If no appropriate production rule is available, then one is left with a piece of prose which may or may not be more elucidating than the sentence one started with, dependent upon the representations in the external world that the the recipient associates withe those sentences. The meaning, if any, of a given sentence is therefore, in general, a function of the recipient.

A particular place, however, is taken by those sentences which are questions to which a precisely defined answer can be given. It is generally understood that anyone who gives the correct answer has "understood" the "meaning" of the question. Therefore, the meaning of a question simply must be the answer to it. We shall show now how the answer to a question belonging to a class of simple questions, concerning elementary-school and secondary-school arithmetic, can be constructed by purely syntactical means.

All knowledge starts with numbers; hence we introduce the metanotion *NUMBER* by means of the following production rules:,

NUMBER : 0 ; SOME . (M6)

SOME : *1* ; *MORE*. (M7)

MORE : *2* ; *MANY*. (M8)

MANY : *HIGH* ; *SOME DIGIT*. (M9)

HIGH : *3* ; *4* ; *5* ; *6* ; *7* ; *8* ; *9*. (M10)

DIGIT : *0* ; *1* ; *2* ; *HIGH*. . (M11)

The productions of *NUMBER* are therefore the ten one-digit numbers *0* up to *9* and, moreover, all more-digit numbers, i.e., all sequences of more than one digits not beginning with *0*, like *365* and *691*.

As a first example of the use of *NUMBER*, we define *word* as follows:

word : *NUMBER letter word*. (N10)

1 letter word : *letter*. (N11)

MORE letter word : *letter* , *MORE minus one letter word*. . (N12)

This definition is incomplete since no production rules for *MORE minus one letter word* are given. We skip this gap for a moment, in order to show a similar situation:

word of at most 1 letter : *1 letter word*. (N13)

word of at most MORE letters : *word of MORE letters* ;

word of less than MORE letters. (N14)

word of less than 2 letters : *word of at most 1 letter*. (N15)

word of less than MANY letters :

word of at most MANY minus one letters. . (N16)

These definitions show a similar deficiency. Of course, our education enables us to interpret the meaning of phrases like *3 minus one letter word* and *word of at most 100 minus one letters* as *2 letter word* and *word of at most 99 letters* respectively and the trouble one has had to acquire this ability may suggest that this interpretation requires the knowledge of semantics. However, we show now how this can be dealt with by syntax only.

We realize that what has to be defined is the predecessor of a given number in a suitably restricted context. This is done by the following hyperrules:

BEGIN 1 minus one END : BEGIN 0 END. (N17)

BEGIN 1 minus one 9 TAIL END : BEGIN 9 TAIL END. (N18)

BEGIN SOME 1 minus one TAIL END : BEGIN SOME 0 TAIL END. (N19)

BEGIN HEAD 2 minus one TAIL END : BEGIN HEAD 1 TAIL END. (N20)

BEGIN HEAD 3 minus one TAIL END : BEGIN HEAD 2 TAIL END. (N21)

BEGIN HEAD 4 minus one TAIL END : BEGIN HEAD 3 TAIL END. (N22)

BEGIN HEAD 5 minus one TAIL END : BEGIN HEAD 4 TAIL END. (N23)

BEGIN HEAD 6 minus one TAIL END : BEGIN HEAD 5 TAIL END. (N24)

BEGIN HEAD 7 minus one TAIL END : BEGIN HEAD 6 TAIL END. (N25)

BEGIN HEAD 8 minus one TAIL END : BEGIN HEAD 7 TAIL END. (N26)

BEGIN HEAD 9 minus one TAIL END : BEGIN HEAD 8 TAIL END. (N27)

BEGIN SOME 0 minus one TAIL END : BEGIN SOME minus one 9 TAIL END. (N28)

The context-restricting metanotions *BEGIN*, *END*, *HEAD* and *TAIL* are produced, for example, by

BEGIN : EMPTY ; TEXT CLOSURE. (M12)

EMPTY : . (M13)

TEXT : CHARACTER ; CHARACTER TEXT. (M14)

CHARACTER : LETTER ; DIGIT ; OPERATOR ; ORDINATOR ; TERMINATOR. (M15)

OPERATOR : − ; + . (M16)

ORDINATOR : , ; ; ; ' : . (M17)

TERMINATOR : . ; ? ; ! . (M18)

CLOSURE : is ; with ; of ; at most ; less than ; OPERATOR. (M19)

END : EMPTY ; TERMINATOR ; NOUN END ; OPERATOR NUMBER END. (M20)

NOUN : letters ; letter word ; stamps. (M21)

HEAD : EMPTY ; SOME. (M22)

TAIL : *EMPTY* ; *9 TAIL*. . (M23)

These productions are by no means exhaustive; they just serve as illustrations and are sufficient for our examples. Note that *BEGIN* stands either for nothing or for an arbitrary text ending with a recognizable closure like *at most* or an operator, but not with a digit. Similarly, *END* stands either for nothing or for some text, here severely restricted for the sake of simplicity, which does not begin with a digit. On the other hand, *HEAD* stands either for nothing or for a number greater than zero, and *TAIL* stands for a, possibly empty, sequence of nines.

Consider now *3 minus one letter word*. By letting *BEGIN*, *HEAD* and *TAIL* stand for nothing and *END* for *letter word*, (N21) takes the form

3 minus one letter word : *2 letter word*. ,

by letting *MORE* stand for *2*, (N12) takes the form

2 letter word : *letter* , *2 minus one letter word*. ,

and by letting *BEGIN*, *HEAD* and *TAIL* stand for nothing and *END* for *letter word*, (N20) takes the form

2 minus one letter word : *1 letter word*. ,

so that one has eventually by virtue of (N11) and (N12)

3 letter word → *letter* , *letter* , *letter*. .

In the same way, we find from (N16), by letting *MANY* stand for *100*,

word of less than 100 letters : *word of at most 100 minus one letters*.

By letting *BEGIN* stand for *word of at most* (i.e., *TEXT* for *word of* and *CLOSURE* for *at most*), *SOME* for *10*, *TAIL* for nothing and *END* for *letters*, (N28) takes the form

word of at most 100 minus one letters :

word of at most 10 minus one 9 letters. .

By letting *BEGIN* stand for *word of at most*, *SOME* for *1*, *TAIL* for *9* and *END*

for *letters*, (N28) takes the form

word of at most 10 minus one 9 letters :

word of at most 1 minus one 99 letters. .

By letting *BEGIN* stand for *word of at most*, *TAIL* for *9* and *END* for *letters*, (N18) takes the form

word of at most 1 minus one 99 letters : *word of at most 99 letters*. .

By letting *MORE* stand for *99*, (N14) takes the form

word of at most 99 letters : *word of 99 letters* ;

word of less than 99 letters.

and (N16) takes the form

word of less than 99 letters : *word of at most 99 minus one letters*. .

In the same way, *word of at most 99 minus one letters* is produced to *word of 98 letters* and *word of at most 98 minus one letters*, and so on.

At last, we turn our attention to complete sentences like

how much is 691 - 365?

and

how much is 365 - 691? .

The answer to the first question is generally thought to require some elementary-school training, not in "reading" or "writing" but in "arithmetic", and the answer to the second question is similarly thought to require some secondary-school training, perhaps in "algebra". In spite of the fact that arithmetic and algebra are usually said to require "understanding", the production of the correct answer to these and similar questions is simply a process in the realm of syntax only.

The concept of subtraction of a number from a second number is defined, using the metanotion *ZOME* produced by

ZOME : *SOME*. , (M24)

by the following three hyperrules:

BEGIN SOME - ZOME END : BEGIN SOME minus one - ZOME minus one END. (N29)

BEGIN NUMBER - 0 END : BEGIN NUMBER END. (N30)

BEGIN 0 - SOME END : BEGIN - SOME END. . (N31)

By letting *BEGIN* stand for *how much is*, *SOME* for *691*, *ZOME* for *365* and *END* for *?*, (N29) takes the form

how much is 691 - 365? : how much is 691 minus one - 365 minus one?. .

By letting *BEGIN* stand for *how much is 691 minus one -* (i.e., *TEXT* for *how much is 691 minus one* and *CLOSURE* for -), *HEAD* for *36*, *TAIL* for nothing and *END* for *?* , (N23) takes the form

how much is 691 minus one - 365 minus one? :

how much is 691 minus one - 364?.

and by letting *BEGIN* stand for *how much is* (i.e., *TEXT* for *how much* and *CLOSURE* for *is*), *SOME* for *69* and *END* for *- 364?* , (N19) takes the form

how much is 691 minus one - 364? : how much is 690 - 364?. .

This process goes on until *how much is 326 - 0?* is produced, which is then produced by (N30) into *how much is 326?* Similarly, the second question, *how much is 365 - 691?* is produced into *how much is - 326?*. Defining the metanotion *SIGN* by

SIGN : EMPTY ; -. , (M25)

we formulate our last hyperrule

how much is SIGN NUMBER : SIGN NUMBER. . (N32)

by means of which eventually

how much is 691 - 365? → 326. .

and

how much is 365 - 691? → - 326. .

are produced.

These examples show that there exists a class of sentences, questions, in the English language whose semantic content, the answers to those questions, can be produced by syntactic means.

References

[1] A. van Wijngaarden, Orthogonal design and description of a formal language, MR 76, Math. Centrum, Amsterdam, 1965.

[2] A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck and C.H.A. Koster, Report on the algorithmic language ALGOL 68, MR 101, Math. Centrum, Amsterdam, 1969.

[3] M. Sintzoff, Existence of a Van Wijngaarden syntax for every recursively enumerable set, Extrait des Annales de la Société Scientifique de Bruxelles, T. '81, II, pp. 115 - 118, 1967.