stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 15/74 FEBRUARY

L.G.L.T. MEERTENS & J.C. van VLIET

REPAIRING THE STATE SWITCHER SKELETON
OF ALGOL 68 PROGRAMS

Prepublication

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

ACM - Computing Reviews - category: 4.12, 4.22

Repairing the State Switcher Skeleton of ALGOL 68 Programs *)

by

L.G.L.T. Meertens and J.C. van Vliet

Summary

This paper deals with one of the concrete aspects of the syntax of ALGOL 68, viz., the parenthesis structure. Upon encountering an error in a piece of source text, good resynchronization of a parser is only then possible if it is known beforehand which opening parentheses are, and which are not, accompanied by a matching closing parenthesis (and vice versa). This holds especially for "state switchers", i.e., parentheses enclosing sequences of symbols that lack syntactical structure (the inside of comments, pragmats and string-denotations), which have one same representation for opening and closing parenthesis. A practical algorithm for repairing incorrect state switcher skeletons is given. This algorithm determines an "admissible interpretation" of maximum likelihood, using dynamic programming.

*) This paper is an extended version of the first part of [1]; moreover, the revision of ALGOL 68 has been taken into account.

This paper is not for review; it is meant for publication in a journal.

Contents

0. Preface

The degree in which compilers for languages like ALGOL 68 [2] are able to recover from errors in the source text and to give meaningful error messages, i.e., error messages which are interpretable for the human programmer, varies considerably in practice. In the current effort undertaken at the Mathematical Centre to construct a machine-independent ALGOL 68 compiler, one of the design objectives is to reach a relatively high level of error-recoverability. This objective sprouts forth from the following two considerations:

(i) It is expected that a major application area for our compiler will be the processing of student programs, where the emphasis is on the correction of syntactical errors. We hope to minimize the number of runs required to make the program syntactically correct.

(ii) The generality of ALGOL 68, both on the abstract and the concrete aspects of syntax, tends to give rise to error messages which are difficult to interpret, unless this tendency is counteracted by a conscious effort.

The present publication deals with one of the concrete aspects of the syntax of ALGOL 68, viz., the parenthesis structure.

If, somewhere in a piece of source text which starts with an opening parenthesis, an error occurs which causes the parser to be derailed, one may hope to use the closing parenthesis to bring it back in its track. Should, however, this closing parenthesis be missing (which might be the cause of derailment in the first place), then this strategy is not particularly helpful. It may appear that the solution would be to insert, as it were, the matching closing parenthesis in the source text when a different closing parenthesis is met, but then, if the source text contains an extra closing parenthesis, we are even worse off. The conclusion is that a good resynchronization of the parser is only then possible if it is known beforehand which opening parentheses are, and which are not, accompanied by a matching closing parenthesis (and vice versa). This holds especially for

some parentheses, such as the quote-symbol, that have one same representation for opening and closing parenthesis. Therefore, it was decided to tackle this point in a radical way: the design of the first scan has been extended with the task to repair incorrect parenthesis skeletons.

1. Introduction

In the sequel, the term "parenthesis" will be used to denote a wider class of symbols than is usual: we shall use this term to stand for the following symbols, or rather, representations of symbols[1]:

"braces": $, (,), *begin*, *end*, [,],
|, |:, *case*, *in*, *ouse*, *out*, *esac*,
if, *then*, *elif*, *else*, *fi*, *for*, *from*,
by, *to*, *while*, *do*, *od*, and

"state switchers": ", ¢, #, *co*, *comment*, *pr*, *pragmat*.

The role played by the state switchers is so special as to warrant a separate treatment. Not only does one same symbol serve both as "opener" and as "closer" of certain constructions (as is also the case with the formatter-symbol), but, which is more important, the "item sequences" which are embraced by these symbols (the inside of comments, pragmats and string-denotations) lack syntactical structure and may contain braces in an arbitrary fashion that otherwise would have to occur "nested". Therefore, it is a hopeless task to treat the braces before it is known which parts of the program are, and which are not, item sequences, and, consequently, which braces have to be disregarded and which have to be taken into account. Moreover, such an item sequence may not contain another similar construction. (E.g., a comment may not contain another comment, although it may, possibly, contain the sequence of comment-items # *a* #, whereas format-texts may contain other format-texts.)

---

[1] Since it is, at this stage, neither possible nor necessary to make a distinction between, e.g., an open-symbol and a brief-begin-symbol or a style-i-sub-symbol, we shall use the paranotion open-symbol to stand for any of those. Moreover, where no confusion can arise, "symbol" will be used to indicate both symbols (in the sense of the ALGOL 68 report), and representations of symbols, indiscriminately.

Errors in the state switcher skeleton are "repaired" by marking a number of state switchers such that, by doubling these state switchers, a correct skeleton is obtained. E.g., the state switcher skeleton

¢ ¢ ¢ # ¢ # ¢

might be repaired by marking it thus:

¢ ¢ ¢* # ¢ # ¢*

A principle to which we have strictly adhered is: no correct program will be "repaired". In fact, an even stronger version of this principle, to be formulated later on, applies to our repairing algorithm.

2. The treatment of state switchers

An ALGOL 68 program can be thought of as consisting of a sequence of (possibly empty) segments, separated by state switchers. To each of these segments a "state" may be assigned, which is either *"N"* (neutral) or Σ_C, where *C* stands for one of the state switchers. For a correct program, it is possible to assign these states in such a way that the first and the last segment are *N* and that at each state switcher *C* we have a correct transition, i.e., the state switches to Σ_C if it was *N* and to *N* if it was Σ_C, and otherwise the state is not affected. To give an example:

segments:	~~~~	"	~~~~	#	~~~~	"	~~~~	¢	~~~~	co	~~~~	"	~~~~	¢	~~~~
states :	*N*		$\Sigma_{"}$		$\Sigma_{"}$		*N*		$\Sigma_{¢}$		$\Sigma_{¢}$		$\Sigma_{¢}$		*N*

(Note that the segments which have Σ_C as state are precisely those segments which are, or are contained in, an item sequence.) Obviously, if such an assignment of states is not possible, the program is incorrect. However, the transition of the state at some state switcher may be (locally) correct.

It is necessary to refine our definition of a correct transition slightly further. Although state switchers have one same representation for openers and closers, it is possible in some cases to derive from the context that a given state switcher, which then must be a quote-symbol, cannot be an opener or a closer. E.g., in the context of *d = "monday";* it can be shown that the first quote-symbol must be an opener and the second one a closer. Now, for a state switcher which has been shown to be a non-opener, the transition from the state *N* to the state $\Sigma_{"}$ is not considered correct. A similar restriction applies to state switchers which have been shown to be non-closers.

The task of the algorithm for correcting the state-switcher skeleton can be formulated approximately as follows: assign states to each of the segments in such a way that the number of incorrect transitions is kept, in

some sense, as low as possible. Before we are able to give a more accurate formulation, it is necessary to indicate what incorrect transitions and what interpretations, i.e., assignments of states to all of the segments, are admissible.

2.1. Admissible incorrect transitions

The elementary repairing actions consist of the marking of one state switcher, indicating that it should be disregarded in order to obtain a correct state-switcher skeleton. This implies that the state should not switch at such a state switcher. Therefore, for an incorrect transition to be admissible in such cases, it is necessary that the state does not change. Now, consider the following example with an obviously incorrect skeleton:

~~~~ " ~~~~ " ~~~~ " ~~~~ .

There exist three ways to repair this skeleton, each giving an interpretation with one incorrect transition.

~~~~ " ~~~~ " ~~~~ "* ~~~~ ;
N $\Sigma_{"}$ *N* *N*

~~~~ " ~~~~ "* ~~~~ " ~~~~ ;
N $\Sigma_{"}$ $\Sigma_{"}$ *N*

~~~~ "* ~~~~ " ~~~~ " ~~~~ .
N *N* $\Sigma_{"}$ *N*

Not each of these interpretations is equally desirable, for the following reason: The effect of assigning the state *N* to a segment is that it will be subject to syntactical analysis. If the corresponding segment in the "intended program" was *N*, this is obviously all right, but if it was

(part of) an item sequence, this will probably give rise to some extraneous error messages. On the other hand, if a state other than *N* is assigned to a segment which in the intended program was *N*, the syntactical analysis of this segment will be omitted, with the possible result that some syntactical errors which otherwise would have been detected, will pass unnoticed, thus necessitating an extra run to detect these errors. In our philosophy this latter eventuality is considered far more undesirable than to require the programmer to ignore some error messages. Consequently, both the first and the last interpretation in our example are preferable to the middle one, where the states of both the left and the right segment in the incorrect transition are not *N*, and which, therefore, will not be admitted.

Another type of incorrect, but admissible, transition is found in those transitions which require a non-opener or a non-closer to be interpreted as an opener or closer, respectively.

The following criterion is obtained: an incorrect transition is admissible if and only if either

(i) the state of both segments concerned is *N*, or

(ii) the state switcher is a non-opener and the state switches from *N* to $\Sigma_{//}$, or

(iii) the state switcher is a non-closer and the state switches from $\Sigma_{//}$ to *N*.

2.2. Admissible interpretations

An interpretation I is "locally admissible" if:

(i) all transitions of I are either correct, or admissible incorrect transitions, and

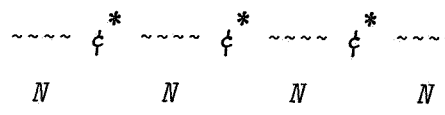
(ii) I assigns the state *N* to the first and the last segment.

An interpretation I is admissible if:

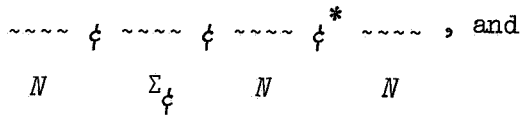
(i) it is locally admissible, and

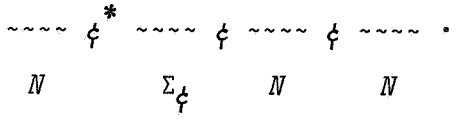
(ii) there does not exist another locally admissible interpretation J, all of whose transitions are either correct, or are the same as the corresponding transition of I.

E.g., although the following interpretation is locally admissible,



it is not admissible, because of the existence of the following two (admissible) interpretations:





Clearly, there always exists at least one admissible interpretation. This can be shown, using the following argument:

(i) There exists at least one locally admissible interpretation, viz., the one which assigns the state *N* to each segment.

(ii) Some locally admissible interpretation not being admissible implies the existence of another locally admissible interpretation with a smaller number of incorrect transitions. Since this number can, obviously, not be less than zero, by *reductio ad absurdum* our claim follows.

If the state-switcher skeleton was correct to begin with, there is only one admissible interpretation, that in which each transition is correct.

2.3. Comparing admissible interpretations

In the general case, there will be more than one admissible interpretation for a given sequence of segments. The problem is, therefore, to give a criterion according to which one of these interpretations can be chosen as, hopefully, the best. A simple criterion would be to count the number of

incorrect transitions. We have chosen, however, for a more sophisticated criterion, mainly because all too often the same number of errors will be found for different interpretations. For example, for the state switcher skeleton

~~~~ ¢ ~~~~ ¢ ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~

there are five admissible interpretations, each of which contains two errors:

(1) ~~~~ ¢ ~~~~ ¢ ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~ #* ~~~~ ¢* ~~~~ ,
N $\Sigma_{¢}$ *N* $\Sigma_{¢}$ $\Sigma_{¢}$ *N* *N* *N*

(2) ~~~~ ¢ ~~~~ ¢ ~~~~ ¢* ~~~~ # ~~~~ ¢ ~~~~ # ~~~~ ¢* ~~~~ ,
N $\Sigma_{¢}$ *N* *N* $\Sigma_{\#}$ $\Sigma_{\#}$ *N* *N*

(3) ~~~~ ¢* ~~~~ ¢ ~~~~ ¢ ~~~~ #* ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~ ,
N *N* $\Sigma_{¢}$ *N* *N* $\Sigma_{¢}$ $\Sigma_{¢}$ *N*

(4) ~~~~ ¢ ~~~~ ¢ ~~~~ ¢* ~~~~ #* ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~ ,
N $\Sigma_{¢}$ *N* *N* *N* $\Sigma_{¢}$ $\Sigma_{¢}$ *N*

(5) ~~~~ ¢* ~~~~ ¢ ~~~~ ¢ ~~~~ # ~~~~ ¢ ~~~~ # ~~~~ ¢* ~~~~
N *N* $\Sigma_{¢}$ *N* $\Sigma_{\#}$ $\Sigma_{\#}$ *N* *N*

Rather than having all incorrect transitions weigh equally heavily, different "error values" have been assigned to the various types of incorrect transitions, based upon estimates of the likelihood of these transitions. Moreover, to some extent the symbols of which the segments consist are taken into account and compared with the state assigned to the segment, the idea
~~~~

being that, e.g., *begin* is more likely to occur in a neutral segment, than in other segments.[2]

In order to obtain estimates of the likelihood, the following, admittedly oversimplified, model has been used: There exists a universe of intended programs, having certain statistical properties (such as the mean length of a neutral segment). A source text is obtained in two steps. First, an intended program is drawn from the universe. Second, this program is subjected to a perturbation process, consisting of omission of marks or of replacement by other marks, at random.[3] Given the a priori distribution of intended programs and the statistical properties of the perturbation process, *Bayesian analysis* [3] makes it possible, for a given source text, to derive the a posteriori probability that the source text was obtained from a certain intended program.

To illustrate the line of thought, a simple example, taken from natural language, may be useful. Take the following sentence:

s1: *A lold mehal was awarmed.*

Obviously, this is meaningless. We may assume something meaningful was intended, but that printing errors have garbled the message. What now was the intended message? There are several possible candidates. Suppose that, after having eliminated some of them, such as:

A bold metal was alarmed. ,

---

[2] The introduction of this type of argument might open the door for the clearly undesirable situation where a correct but very unlikely piece of program is "repaired" into a more likely one. This is precluded, however, by our definition of "admissible interpretation". The obvious advantage is the possibility to resynchronize when, for example, one comment-symbol has disappeared in a piece of program richly supplied with comments.

[3] "Marks" are the atomic elements from which representations are built up, for example, the characters of a given character set.
The case of insertion of marks is not considered here, since the resulting symptoms can be ascribed to the (much more likely) replacement. E.g., the intended program corresponding to the source text *(1¢)* could have been *(1)*, but could equally well have been *(10)*, *(11)*, etc.
Likewise, the case of interchanging marks is not taken into consideration. Such errors rarely influence the parenthesis skeleton; also, they give rise to symptoms, if any, that can be explained in terms of replacement.

on the ground of their being ungrammatical or meaningless, we are left with two candidates:

s2: *A cold meal was warmed.* , and

s3: *A gold medal was awarded.*

Now suppose that we know the a priori probability, presumably based on the context, that the intended message was s2 or, alternatively, s3. We shall denote these by P(s2) and P(s3), respectively. If we also know the probabilities P(s1|s2) and P(s1|s3) that, due to printing errors, s2 and s3, respectively, will be turned into s1, we can compute the a posteriori probabilities that s2 or s3 were intended, given the fact that we have received s1, by the formulae:

$$P(s2|s1) = \frac{P(s1|s2)\ P(s2)}{P(s1|s2)\ P(s2) + P(s1|s3)\ P(s3)} \text{ , and}$$

$$P(s3|s1) = \frac{P(s1|s3)\ P(s3)}{P(s1|s2)\ P(s2) + P(s1|s3)\ P(s3)} .$$

P(s1|s2) and P(s1|s3) can be determined by taking the product of the conditional probabilities for the individual printing errors (which are considered independent). So

$$P(s1|s2) = P(c \to l)\ P(+\ h)\ P(+\ a), \text{ and}$$

$$P(s1|s3) = P(g \to l)\ P(d \to h)\ P(d \to m).$$

The latter probabilities may be estimated by collecting statistical data on printing errors. Observe that, typically, the formulae for P(s2|s1) and P(s3|s1) have their denominators in common, so that, in order to compare them, it suffices to compare their numerators.

In our situation, we have no a priori ground to consider one intended program more likely than another one. Therefore, we will assume a uniform distribution of the intended programs[4]. For the above example, this would imply P(s2) = P(s3). Again, as our goal is comparison only, we obtain a simplification: it now suffices to compare the product of the conditional probabilities for the individual phenomena observed. Rather than multiplying probabilities and comparing products, we shall add error values, i.e., (scaled) logarithms of the probabilities and then compare the sum.

2.4. Determination of error values

As the model on which our algorithm is based is a gross simplification, no other significance should be attached to the error values obtained than that of a heuristic guide, obtained by making educated guesses.

In the subsequent sections, the following will be used to stand for (estimates of) probabilities and other parameters:

N = size of the mark set,

L = size of the letter set,

ϕ_- = probability that a given mark is omitted in the perturbation process,

ϕ_0 = probability that a given mark is replaced by some other mark,

S_Σ = the range over which the length of segments with state Σ may vary in practice, where Σ stands for some state switcher,

---

[4] This is not wholly true, as we want to consider a program containing, e.g., the string *"begin"* less likely than the program with that string replaced by *"begin"*. By means of a trick, however, we can save the uniform distribution. This can be described in terms of our model by restricting the universe of intended programs to programs that do not contain such strings, and than having these strings appear in the perturbation process.

2.4.1. Error values for comment-symbols

Where in the sequel C is used, this should be understood as standing for any of the state switchers ¢, #, *co*, *comment*, *pr* or *pragmat*.

Consider two segments with states Σ_1 and Σ_2, separated by C:

$$\underset{\Sigma_1}{\sim\sim\sim\sim}\ C\ \underset{\Sigma_2}{\sim\sim\sim\sim}$$

In order that an interpretation which assigns Σ_1 and Σ_2 to these segments be admissible, it is necessary that at C we have either a correct or an admissible incorrect transition. So we must have one of the following (classes of) transitions:

(i) $\Sigma_1 = N$, $\Sigma_2 = \Sigma_C$.

(ii) $\Sigma_1 = \Sigma_C$, $\Sigma_2 = N$.

(iii) $\Sigma_1 = \Sigma_2 = \Sigma$, where $\Sigma \neq \Sigma_C$ and $\Sigma \neq N$.

(iv) $\Sigma_1 = \Sigma_2 = N$.

It now remains to be seen how these cases can arise by subjecting an intended program to the perturbation process. In transitions (i), (ii), and (iii) the intended program was not affected, as the transition is correct there. Although the probability of this phenomenon is slightly less than 1, as there is a small, but finite, probability that some perturbation will occur, we shall equate this probability to 1. In case (iv), an incorrect transition is involved, and there is a multitude of possibly intended programs that could result in this phenomenon. In order to restrict the analysis to local effects, we shall assume that this incorrect transition is the only one in the interpretation of the source text. The given source text could have resulted from the replacement of some mark in the intended program by C, but in the light of the much likelier possibility that the corresponding C was omitted or mistyped, we shall disregard this possibility. So the class of possibly intended programs comprises those programs that can be transformed into the source text by deleting one occurrence of C, or replacing it by some other

mark (more specifically, by the corresponding mark in the source text). For each of these, we have: $P = \phi_- + \frac{\phi_0}{N}$. The number of possibly intended programs can be estimated roughly to be $2S_C$, one for each of the about S_C places to the left and to the right of C in the source text where the deleted or replaced C could have stood.

To summarize:

$$P(i) = P(ii) = P(iii) = 1;$$

$$P(iv) = 2S_C(\phi_- + \frac{\phi_0}{N}).$$

The corresponding error values are obtained by taking the logarithm to some base of these probabilities.

2.4.2. Error values for quote-symbols

Previously, we have already mentioned the fact that it is sometimes possible to see whether a given quote-symbol is an opener or a closer. As we expect that in most cases string-denotations will consist of letters mainly, only the information provided by these symbols is taken into account. String-denotations may only be followed and/or preceded by letters when they occur within format-texts. Note that at this stage it is not yet known whether a given string-denotation does or does not occur in a format-text: this can only be checked when the brace skeleton is known, which in its turn has to await the treatment of the state switcher skeleton. The set of letters which may occur in format-texts consists of: *a*, *b*, *c*, *d*, *e*, *f*, *g*, *i*, *k*, *l*, *n*, *p*, *q*, *r*, *s*, *t*, *x*, *y* and *z*. In order to make the probability of recognizing quote-symbols as openers or closers as great as possible, we do not only consider one symbol, but a maximal sequence of letters, i.e., a sequence of letters preceded and followed by a non-letter. Let $\bar{\delta}$ stand for the set of letters mentioned above, and δ for the set of remaining letters, *h*, *j*, *m*, *o*, *u*, *v* and *w*. Then a $\bar{\delta}$-sequence is a, possibly empty, maximal sequence of letters, each letter belonging to $\bar{\delta}$. A δ-sequence is a maximal sequence of letters, at least one of which is an element of δ. (So a sequence of letters is either a $\bar{\delta}$-sequence or a δ-sequence.) To give an example: *asnailspace* is a $\bar{\delta}$-sequence, whereas *andromeda* is not. (If a

space-symbol has some representation, e.g., $\dot{-}$, it should of course also be included in δ.) In the sequel, Δ will stand for a δ-sequence, and $\bar{\Delta}$ for a $\bar{\delta}$-sequence. In this way, we can distinguish four cases, depending on the context of the quote-symbol: $\bar{\Delta}$ " Δ, Δ " $\bar{\Delta}$, $\bar{\Delta}$ " $\bar{\Delta}$ and Δ " Δ.

The case $\bar{\Delta}$ " Δ

As in section 2.4.1., we must have one of the four (classes of) transitions (i) to (iv) (where *C* stands for ").

Transition (i) and (iii) involve no errors, so P(i) = P(iii) = 1.

Transition (ii) can arise by mutilation of an intended program where the context of the quote-symbol is $\bar{\Delta}$ " $\bar{\Delta}$. The $\bar{\Delta}$ at the right must be an empty sequence, except within format-texts, but for the sake of simplicity we shall disregard the latter case. The change of $\bar{\Delta}$ to Δ can occur by:

- replacing the mark after the quote-symbol by the letter from δ ($P = (N-L)\ \frac{\phi_0}{N}$);
- omitting the mark after the quote-symbol ($P = (N-L)\ \phi_-$).

Taken together, $P(ii) = (N-L)(\phi_- + \frac{\phi_0}{N})$.

Transition (iv) can be treated as in 2.4.1., but now there are only about $S_{"}$ places where the original quote-symbol could have stood since, clearly, the quote-symbol at hand is an opener.

To summarize: $P(i) = P(iii) = 1$;

$$P(ii) = (N-L)\ (\phi_- + \frac{\phi_0}{N});$$

$$P(iv) = S_{"}\ (\phi_- + \frac{\phi_0}{N}).$$

The case Δ " $\bar{\Delta}$

This case is the mirror image of the case $\bar{\Delta}$ " Δ.

We obtain: $P(ii) = P(iii) = 1$;

$$P(i) = (N-L)\ (\phi_- + \frac{\phi_0}{N});$$

$$P(iv) = S_{"}\ (\phi_- + \frac{\phi_0}{N}).$$

The case $\bar{\Delta}$ " $\bar{\Delta}$

This case is similar to the case treated in section 2.4.1., if we take *C* to stand for ".

We have: $P(i) = P(ii) = P(iii) = 1;$

$$P(iv) = 2S_{"}\ (\phi_{-} + \frac{\phi_0}{N}).$$

The case Δ " Δ

This is only correct if transition (iii) is involved. For transition (ii), the analysis of the case $\bar{\Delta}$ " Δ applies, as does that of the case Δ " $\bar{\Delta}$ for transition (i). In the remaining case of transition (iv), the intended program must have been one where instead of the quote-symbol at hand some other symbol stood ($P = N\phi_0 \frac{1}{N} = \phi_0$).

Summarizing: $P(i) = P(ii) = (N-L)\ (\phi_{-} + \frac{\phi_0}{N});$

$$P(iii) = 1;$$

$$P(iv) = \phi_0.$$

2.4.3. Numerical results

Guesstimates for the various parameters in this section were obtained independently from some colleagues and used to compute the various error values (where the negative logarithm to some base was taken). The results are listed in table 1. The error value for bold symbols in a non-neutral segment was, after some experimentation, chosen equal to 5.

2.5. The algorithm

The task of the algorithm can be stated thus: find among all admissible interpretations an optimal one, i.e., one with minimal total error value for the segments and transitions involved. Obviously, it is impractical to generate all admissible interpretations one by one, as their number will grow exponentially with the number of state switchers in the source text. By applying the principle of *dynamic programming* [4], however, it is possible to derive a practical algorithm.

Table 1. Error values.

Estimates:

N	64	64	64
L	27	27	27
ϕ_-	.0001	.00163	.0003
ϕ_0	.0002	.00163	.0020
S_C	20	50	100
$S_{''}$	20	17	10

Error values

C	transition				average
comment					
	i,ii,iii	0	0	0	0
	iv	11.3	6.8	7.0	8
$\bar{\Delta}$ " Δ					
	i,iii	0	0	0	0
	ii	11.5	10.5	11.4	11
	iv	12.7	13.4	14.8	14
Δ " $\bar{\Delta}$					
	i	11.5	10.5	11.4	11
	ii,iii	0	0	0	0
	iv	12.7	13.4	14.8	14
$\bar{\Delta}$ " $\bar{\Delta}$					
	i,ii,iii	0	0	0	0
	iv	11.3	10.8	13.0	12
Δ " Δ					
	i,ii	11.5	10.5	11.4	11
	iii	0	0	0	0
	iv	17.6	24.2	16.2	19

i: $N \rightarrow \Sigma_C$

ii: $\Sigma_C \rightarrow N$

iii: $\Sigma_{C'} \rightarrow \Sigma_{C'}$ $(C' \neq \Sigma_C$ or $N)$

iv: $N \rightarrow N$

2.5.1. Application of the dynamic programming principle

Let the segments of a source text be numbered from 0 to n. Consider two locally admissible interpretations I and I' that assign the same state Σ_i to the i-th segment. So we have

$$I : \underset{\Sigma_0}{\overset{(0)}{\sim\sim\sim}} T_1 \underset{\Sigma_1}{\overset{(1)}{\sim\sim\sim}} \cdots \underset{\Sigma_{i-1}}{\overset{(i-1)}{\sim\sim\sim}} T_i \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} T_{i+1} \underset{\Sigma_{i+1}}{\overset{(i+1)}{\sim\sim\sim}} \cdots \underset{\Sigma_{n-1}}{\overset{(n-1)}{\sim\sim\sim}} T_n \underset{\Sigma_n}{\overset{(n)}{\sim\sim\sim}} \text{ and}$$

$$I' : \underset{\Sigma'_0}{\overset{(0)}{\sim\sim\sim}} T'_1 \underset{\Sigma'_1}{\overset{(1)}{\sim\sim\sim}} \cdots \underset{\Sigma'_{i-1}}{\overset{(i-1)}{\sim\sim\sim}} T'_i \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} T'_{i+1} \underset{\Sigma'_{i+1}}{\overset{(i+1)}{\sim\sim\sim}} \cdots \underset{\Sigma'_{n-1}}{\overset{(n-1)}{\sim\sim\sim}} T'_n \underset{\Sigma'_n}{\overset{(n)}{\sim\sim\sim}} ,$$

where the T's stand for the transitions involved and the numbers of the segments are given between parentheses. Then, clearly, both of the following interpretations are also locally admissible:

$$\underset{\Sigma_0}{\overset{(0)}{\sim\sim\sim}} T_1 \underset{\Sigma_1}{\overset{(1)}{\sim\sim\sim}} \cdots \underset{\Sigma_{i-1}}{\overset{(i-1)}{\sim\sim\sim}} T_i \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} T'_{i+1} \underset{\Sigma'_{i+1}}{\overset{(i+1)}{\sim\sim\sim}} \cdots \underset{\Sigma'_{n-1}}{\overset{(n-1)}{\sim\sim\sim}} T'_n \underset{\Sigma'_n}{\overset{(n)}{\sim\sim\sim}} \text{ and}$$

$$\underset{\Sigma'_0}{\overset{(0)}{\sim\sim\sim}} T'_1 \underset{\Sigma'_1}{\overset{(1)}{\sim\sim\sim}} \cdots \underset{\Sigma'_{i-1}}{\overset{(i-1)}{\sim\sim\sim}} T'_i \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} T_{i+1} \underset{\Sigma_{i+1}}{\overset{(i+1)}{\sim\sim\sim}} \cdots \underset{\Sigma_{n-1}}{\overset{(n-1)}{\sim\sim\sim}} T_n \underset{\Sigma_n}{\overset{(n)}{\sim\sim\sim}} .$$

In words: it is admissible to cross over at the i-th segment.

To give an example: from the two locally admissible interpretations

$$\underset{N}{\overset{(0)}{\sim\sim\sim}} \text{¢} \underset{\Sigma_{\text{¢}}}{\overset{(1)}{\sim\sim\sim}} \text{¢} \underset{N}{\overset{(2)}{\sim\sim\sim}} \text{¢}^* \underset{N}{\overset{(3)}{\sim\sim\sim}} \#^* \underset{N}{\overset{(4)}{\sim\sim\sim}} \text{¢} \underset{\Sigma_{\text{¢}}}{\overset{(5)}{\sim\sim\sim}} \# \underset{\Sigma_{\text{¢}}}{\overset{(6)}{\sim\sim\sim}} \text{¢} \underset{N}{\overset{(7)}{\sim\sim\sim}} ,$$

and

$$\underset{N}{\overset{(0)}{\sim\sim\sim}} \text{¢}^* \underset{N}{\overset{(1)}{\sim\sim\sim}} \text{¢} \underset{\Sigma_{\text{¢}}}{\overset{(2)}{\sim\sim\sim}} \text{¢} \underset{N}{\overset{(3)}{\sim\sim\sim}} \# \underset{\Sigma_{\#}}{\overset{(4)}{\sim\sim\sim}} \text{¢} \underset{\Sigma_{\#}}{\overset{(5)}{\sim\sim\sim}} \# \underset{N}{\overset{(6)}{\sim\sim\sim}} \text{¢}^* \underset{N}{\overset{(7)}{\sim\sim\sim}} ,$$

which both assign *N* to the third segment, we can derive yet two other locally admissible interpretations:

$$\underset{N}{\overset{(0)}{\sim\sim\sim}} \not\in \underset{\Sigma_{\not\in}}{\overset{(1)}{\sim\sim\sim}} \not\in \underset{N}{\overset{(2)}{\sim\sim\sim}} \overset{*}{\not\in} \underset{N}{\overset{(3)}{\sim\sim\sim}} \# \underset{\Sigma_{\#}}{\overset{(4)}{\sim\sim\sim}} \not\in \underset{\Sigma_{\#}}{\overset{(5)}{\sim\sim\sim}} \# \underset{N}{\overset{(6)}{\sim\sim\sim}} \overset{*}{\not\in} \underset{N}{\overset{(7)}{\sim\sim\sim}} ,$$

and

$$\underset{N}{\overset{(0)}{\sim\sim\sim}} \overset{*}{\not\in} \underset{N}{\overset{(1)}{\sim\sim\sim}} \not\in \underset{\Sigma_{\not\in}}{\overset{(2)}{\sim\sim\sim}} \not\in \underset{N}{\overset{(3)}{\sim\sim\sim}} \overset{*}{\#} \underset{N}{\overset{(4)}{\sim\sim\sim}} \not\in \underset{\Sigma_{\not\in}}{\overset{(5)}{\sim\sim\sim}} \# \underset{\Sigma_{\not\in}}{\overset{(6)}{\sim\sim\sim}} \not\in \underset{N}{\overset{(7)}{\sim\sim\sim}} .$$

This is illustrated perhaps more vividly in the diagram of fig. 1,*) in which the interpretations are given by the paths from left to right. (This way of looking at interpretations has proved a powerful heuristic guide.)

Now consider two partial interpretations

$$P_i: \underset{\Sigma_0}{\overset{(0)}{\sim\sim\sim}} T_1 \underset{\Sigma_1}{\overset{(1)}{\sim\sim\sim}} \ldots \underset{\Sigma_{i-1}}{\overset{(i-1)}{\sim\sim\sim}} T_i \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} \text{ and}$$

$$P_i': \underset{\Sigma_0'}{\overset{(0)}{\sim\sim\sim}} T_1' \underset{\Sigma_1'}{\overset{(1)}{\sim\sim\sim}} \ldots \underset{\Sigma_{i-1}'}{\overset{(i-1)}{\sim\sim\sim}} T_i' \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim}} ,$$

having their last states in common. Suppose that C is an optimal continuation of P_i resulting in a locally admissible interpretation, and, similarly, that C' is an optimal continuation of P_i'. As stated above, C' must be a locally admissible continuation of P_i also, and C of P_i', so we have:

error value (P_iC) $\leq$ error value (P_iC') and

error value $(P_i'C_i')$ $\leq$ error value $(P_i'C)$.

Since the error value is obtained by adding together the error values of the individual transitions and segments, it is possible to apply the principle of dynamic programming. The above inequalities can be written thus:

error value (P_i) + error value (C) $\leq$ error value (P_i) + error value (C')

and

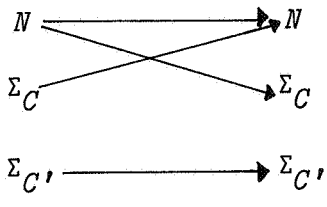
error value (P_i') + error value (C') $\leq$ error value (P') + error value (C).

From this, we derive: error value (C) = error value (C'). Consequently, error value (P_iC) < error value $(P_i'C')$ $\Leftrightarrow$ error value (P_i) < error value (P_i').

*) All figures are collected at the end of this paper.

In words: a partial interpretation P_i can only then beat another partial interpretation P_i', i.e., can only then be the initial part of an interpretation which is better than the best possible completion of P_i', if its error value is less than that of P_i'. As a consequence, only one out of P_i and P_i' needs to be retained. This means that the set of partial interpretations of the segments 0 to i need contain at most one element for each of the (at most) eight possible states Σ_i for the i-th segment, to wit, an optimal partial interpretation. Initially, for i = 0, this set contains only one element, $\underset{\sim\sim\sim\sim}{\overset{(0)}{N}}$. Scanning the state switcher skeleton from left to right, at each state switcher the set of partial interpretations is replaced by a new set. The transitions involved upon encountering a state switcher C can be depicted thus:

$\underset{\sim\sim\sim\sim}{\overset{(i-1)}{\Sigma_{i-1}}}$ $\quad C \quad$ $\underset{\sim\sim\sim\sim}{\overset{(i)}{\Sigma_i}}$



$(\Sigma_{C'} \neq \Sigma_C)$

A decision has to be taken whether the new partial interpretation corresponding to the state N is the continuation of the old partial interpretation corresponding to N $(P_{i-1}(N))$ or of that corresponding to Σ_C $(P_{i-1}(\Sigma_C))$. This can be decided by comparing the sum of the error values of the old partial interpretation and the transition, i.e., by comparing

error value $(P_{i-1}(N))$ + error value $(N \overset{C}{\rightarrow} N)$ and
error value $(P_{i-1}(\Sigma_C))$ + error value $(\Sigma_C \overset{C}{\rightarrow} N)$.

After having processed the last segment of the source text, the (then complete) interpretation corresponding to the state N is an optimal interpretation. To give an example, which is slightly simplified in that only three states are involved, the state switcher skeleton which we already encountered in section 2.3. is displayed in fig. 2 (we assume that in segments 2, 3, 6 and 7 bold symbols occur), together with the correct or admissible incorrect transitions (drawn as arrows between states) and the corresponding error values. Where two transitions come together, one of them is drawn as an arrow with a white head, indicating that the partial interpretation corresponding to

the other transition (drawn with a black head) is more profitable. The partial error values are displayed in boxes. By following the black-headed arrows back from the final state N, the optimal interpretation is obtained.

There is, however, one proviso. Although the interpretation obtained is certainly optimal among the locally admissible interpretations, there is no guarantee at all that, as a whole, it is admissible. Therefore, it is necessary to slightly amend the process sketched above.

2.5.2. Σ-admissibility

We define the notion "Σ-admissible" for partial interpretations in the following way:

Let P_i be a partial interpretation, assigning states to the segments 0 to i. P_i is Σ-admissible (for short: $P_i \in Adm_i(\Sigma)$) if

(i) all transitions of P_i are either correct, or admissible incorrect transitions,

(ii) P_i assigns the state N to segment 0 and Σ to segment i, and

(iii) there does not exist another partial interpretation Q_i satisfying (i) and (ii), all of whose transitions are either correct, or are the same as the corresponding transition of P_i. (We say that such a Q_i "rules out" P_i.)

We can observe some facts, expressed in the form of a

Lemma:

(a) For a (total) interpretation I, the notions "admissible" and "N-admissible" coincide. Consequently, $Adm_n(N) \neq \emptyset$.

(b1) $Adm_0(N) = \left\{ \underset{N}{\overset{(0)}{\sim\sim\sim\sim}} \right\}$.

(b2) For $\Sigma \neq N$, $Adm_0(\Sigma) = \emptyset$.

(c) If $Q_{i-1}(\Sigma')\ T_i\ \underset{\Sigma}{\overset{(i)}{\sim\sim\sim}} \in Adm_i(\Sigma)$, then $Q_{i-1}(\Sigma') \in Adm_{i-1}(\Sigma')$.

(d) If P_i is ruled out from Σ-admissibility by some Q_i (not necessarily Σ-admissible itself), then there exists a Σ-admissible R_i ruling out P_i.

(e) If $P_{i-1}(\Sigma') \in Adm_{i-1}(\Sigma')$ and T_i is a correct transition in the context of $\underset{\Sigma'}{\overset{(i-1)}{\sim\sim\sim\sim}}\, T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$, then $P_{i-1}(\Sigma')\, T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}} \in Adm_i(\Sigma)$.

Proof of (c):

From any R_{i-1} ruling out $Q_{i-1}(\Sigma')$ we can construct $R_{i-1}T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$, ruling out $Q_{i-1}(\Sigma')\, T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$.

Proof of (d):

First, observe that "to rule out" is a transitive relationship. Moreover, if Q_i rules out P_i, then the number of incorrect transitions in Q_i is less than in P_i. Therefore, a sequence of partial interpretations P_i, Q_i, Q_i', Q_i'', ..., each ruled out by the next, must have a finite length (in fact, cannot exceed the number of incorrect transitions in P_i by more than one). Consider such a sequence of <u>maximal</u> length P_i, Q_i, Q_i', Q_i'', ..., $Q_i^{(n)}$ (that is, no $Q_i^{(n+1)}$ exists ruling out $Q_i^{(n)}$). Then, $Q_i^{(n)}$ is such an R_i as we were looking for.

Proof of (e):

Suppose $P_{i-1}(\Sigma')\, T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}} \notin Adm_i(\Sigma)$, so that some $Q_{i-1}(\Sigma'')\, T_i' \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$ rules out $P_{i-1}(\Sigma')\, T_i \,\underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$. Now, assume $T_i \neq T_i'$. From the definition of "ruling out" we derive the correctness of T_i'. So we have $T_i \neq T_i'$, T_i is correct and T_i' is correct, simultaneously. However, for two different transitions to yield one and the same new state Σ, it is necessary that one of them be a transition $N \to N$ and therefore incorrect, which yields a contradiction. Evidently, the assumption was incorrect: $T_i = T_i'$. But then, $Q_{i-1}(\Sigma'')$ alone already rules out $P_{i-1}(\Sigma')$, contradicting $P_{i-1}(\Sigma') \in Adm_{i-1}(\Sigma')$.

2.5.3. The Companion Theorem

Given a partial interpretation $P_i(\Sigma)$ and some state Σ^*, we can informally define the "companion interpretation" of $P_i(\Sigma)$ corresponding to Σ^*

thus: follow, in a diagram in which $P_i(\Sigma)$ is given by a path and in which the correct transitions are indicated, the correct transitions backwards starting from the state Σ^* at the i-th segment, until $P_i(\Sigma)$ is met, and then continue along $P_i(\Sigma)$. The partial interpretation thus obtained is the companion interpretation sought for. For example, in the diagram of fig. 3, in which an interpretation $P_6(\Sigma_f)$ is indicated by a path of bold arrows, we can obtain its companion interpretations corresponding to N and $\Sigma_\#$, as displayed in the diagram of fig. 4. Such a companion interpretation need not always exist, since following the correct transitions backwards may either bring one to a point to which no correct transition leads, or the path thus obtained may never meet the given partial interpretation.

Furthermore, it is possible to distinguish two types of companion interpretations, in that one may or may not require that the last transition of the companion interpretation is a correct one. (This gives a distinction only if $\Sigma = \Sigma^*$ and the last transition of $P_i(\Sigma)$ is incorrect.) It is possible to give a more formal definition, by means of mutual recursion:

$$\mathrm{com}_{\Sigma^*}(P_i(\Sigma)) = \begin{cases} P_i(\Sigma) \text{ if } \Sigma^* = \Sigma \\ \\ \mathrm{Com}_{\Sigma^*}(P_i(\Sigma)) \text{ otherwise} \end{cases},$$

$$\mathrm{Com}_{\Sigma^*}(P_i(\Sigma)) = \begin{cases} \text{if } P_i(\Sigma) = P_{i-1}(\Sigma')\ T_i\ \underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}} \text{ and there exists a} \\ \Sigma^{**} \text{ -which then is uniquely determined- such} \\ \text{that } T_i^* \text{ is a correct transition in the context} \\ \text{of } \underset{\Sigma^{**}}{\overset{(i-1)}{\sim\sim\sim\sim}}\ T_i^*\ \underset{\Sigma^*}{\overset{(i)}{\sim\sim\sim\sim}}, \text{then } \mathrm{com}_{\Sigma^{**}}(P_{i-1}(\Sigma'))\ T_i^*\ \underset{\Sigma^*}{\overset{(i)}{\sim\sim\sim\sim}}, \\ \text{and, otherwise, undefined.} \end{cases}$$

It is now possible to state the

Companion Theorem:

Let $P_i(\Sigma) = P_{i-1}(\Sigma')\ T_i\ \underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$, where $P_{i-1}(\Sigma') \in Adm_{i-1}(\Sigma')$. Then $P_i(\Sigma) \notin Adm_i(\Sigma)$ if and only if $Com_\Sigma(P_i(\Sigma))$ is defined and T_i is an incorrect transition, in which case, moreover, $Com_\Sigma(P_i(\Sigma))$ rules out $P_i(\Sigma)$.

Proof:

Let $P_i(\Sigma)$ be written as

$$\underset{\Sigma_0}{\overset{(0)}{\sim\sim\sim\sim}}\ T_1\ \underset{\Sigma_1}{\overset{(1)}{\sim\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{i-1}}{\overset{(i-1)}{\sim\sim\sim\sim}}\ T_i\ \underset{\Sigma_i}{\overset{(i)}{\sim\sim\sim\sim}}\ ,\ \text{where } \Sigma_{i-1} = \Sigma' \text{ and } \Sigma_i = \Sigma.$$

(if) Suppose $Com_\Sigma(P_i(\Sigma))$ is defined and T_i is an incorrect transition. $Com_\Sigma(P_i(\Sigma))$ may be written as

$$\underset{\Sigma_0^*}{\overset{(0)}{\sim\sim\sim\sim}}\ T_1^*\ \underset{\Sigma_1^*}{\overset{(1)}{\sim\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{i-1}^*}{\overset{(i-1)}{\sim\sim\sim\sim}}\ T_i^*\ \underset{\Sigma_i^*}{\overset{(i)}{\sim\sim\sim\sim}}\ .$$

$\Sigma_i^* = \Sigma_i$, but $T_i^* \neq T_i$, since, by the definition of Com, T_i^* is a correct transition. Consequently, $\Sigma_{i-1}^* \neq \Sigma_{i-1}$. On the other hand, $\Sigma_0^* = \Sigma_0 (= N)$, so there must exist a maximal m, $0 \leq m < i$, such that $\Sigma_m^* = \Sigma_m$. From the definition of Com (and com) we have that for $n \leq m$, $T_n^* = T_n$, and for $n > m$, T_n^* is correct. So $P_i(\Sigma)$ is ruled out by $Com_\Sigma(P_i(\Sigma))$.

(only if) Let the partial interpretation ruling out $P_i(\Sigma)$ be written as

$$\underset{\Sigma_0^*}{\overset{(0)}{\sim\sim\sim\sim}}\ T_1^*\ \underset{\Sigma_1^*}{\overset{(1)}{\sim\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{i-1}^*}{\overset{(i-1)}{\sim\sim\sim\sim}}\ T_i^*\ \underset{\Sigma_i^*}{\overset{(i)}{\sim\sim\sim\sim}}\ .$$

$\Sigma_i^* = \Sigma_i$, but $\Sigma_{i-1}^* \neq \Sigma_{i-1}$, since, otherwise, $P_{i-1}(\Sigma')$ would be ruled out by

$$\underset{\Sigma_0^*}{\overset{(0)}{\sim\sim\sim\sim}}\ T_1^*\ \underset{\Sigma_1^*}{\overset{(1)}{\sim\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{i-1}^*}{\overset{(i-1)}{\sim\sim\sim\sim}}\ .$$

Consequently, $T_i^* \neq T_i$. On the other hand, $\Sigma_0^* = \Sigma_0 (= N)$, so there must exist a maximal m, $0 \leq m < i$, such that $\Sigma_m^* = \Sigma_m$. From (e) of the Lemma we derive the incorrectness of T_i. On the other hand, according to the definition of

ruling out, T_i^* must be correct. Also, for $m < n < i$, T_n^* must be correct, since, otherwise, we would have $T_n^* = T_n$ and, therefore, $\Sigma_n^* = \Sigma_n$, contradicting the maximality of m. Therefore, the following partial interpretation

$$\underset{\Sigma_0}{\overset{(0)}{\sim\sim\sim}}\ T_1\ \underset{\Sigma_1}{\overset{(1)}{\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{m-1}}{\overset{(m-1)}{\sim\sim\sim}}\ T_m\ \underset{\Sigma_m=\Sigma_m^*}{\overset{(m)}{\sim\sim\sim}}\ T_{m+1}^*\ \underset{\Sigma_{m+1}^*}{\overset{(m+1)}{\sim\sim\sim}}\ \cdots\ \underset{\Sigma_{i-1}^*}{\overset{(i-1)}{\sim\sim\sim}}\ T_i^*\ \underset{\Sigma_i^*}{\overset{(i)}{\sim\sim\sim}}$$

serves equally well to rule out $P_i(\Sigma)$. But this is exactly the companion interpretation $Com_\Sigma(P_i(\Sigma))$.

2.5.4. Ensuring admissibility

We are now in a position to describe the amendment to the process: When it has to be decided for some $P_i(\Sigma)$ which of two candidates $P_{i-1}(\Sigma')\ T_i\ \overset{(i)}{\sim\sim}_\Sigma$ and $P_{i-1}(\Sigma'')\ T_i'\ \overset{(i)}{\sim\sim}_\Sigma$ is chosen, then, before their error values are compared, a test is made to see if one candidate, say C, is ruled out by $Com_\Sigma(C)$, in which case only the other candidate is retained. We shall show that, for each i from 0 to n, the set of partial interpretations of the segments 0 to i thus obtained contains for each state Σ a partial interpretation $P_i(\Sigma) \in Adm_i(\Sigma)$, provided of course that $Adm_i(\Sigma) \neq \emptyset$.

It has to be shown:

(A) that, for $i = 0$, the set of partial interpretations fulfils the above requirement and

(B) that, for $i \geq 1$, it is possible to construct from such a set $\{P_{i-1}(\Sigma)\}$ a new set $\{P_i(\Sigma)\}$, also fulfilling the requirement, such that all of its elements are continuations of some element of $\{P_{i-1}(\Sigma)\}$, i.e., if $Adm_i(\Sigma) \neq \emptyset$, then $P_{i-1}(\Sigma')\ T_i\ \overset{(i)}{\sim\sim}_\Sigma \in Adm_i(\Sigma)$ for some Σ' and some transition T_i.

From (A) and (B), together with the Companion Theorem, the claim then follows. For suppose that $P_i(\Sigma)$ has been constructed by the process as the one and only continuation of some $P_{i-1}(\Sigma')$ leading to the state Σ. According to (B), some Σ-admissible continuation exists; so, as there is only one continuation, it must be Σ-admissible. If, on the other hand, $P_i(\Sigma)$ has been chosen from two candidates, then these have been tested against their companion interpretation Com_Σ, so that, if they have been retained, they are, according to the Companion Theorem, Σ-admissible.

Proof of (A):

We have to show: if $Adm_0(\Sigma) \neq \emptyset$, then $P_0(\Sigma) \in Adm_0(\Sigma)$. This follows directly from the initial value of the set of partial interpretations, $\left\{ \underset{\sim\sim\sim\sim}{(0)} \right\}$, combined with (b1) and (b2) of the Lemma.

Proof of (B):

Assume that, for some Σ, $Adm_i(\Sigma) \neq \emptyset$. $Adm_i(\Sigma)$ contains at least one element, which can be written as $Q(\Sigma')\ T_i \underset{\Sigma}{\sim\sim\sim\sim}^{(i)}$. From (c) of the Lemma, we see that $Q(\Sigma') \in Adm_{i-1}(\Sigma')$. $Adm_{i-1}(\Sigma') \neq \emptyset$, so there exists, by hypothesis, a $P_{i-1}(\Sigma') \in Adm_{i-1}(\Sigma')$. If $P_{i-1}(\Sigma')\ T_i \underset{\Sigma}{\sim\sim\sim\sim}^{(i)} \in Adm_i(\Sigma)$, we are done. Suppose therefore $P_{i-1}(\Sigma')\ T_i \underset{\Sigma}{\sim\sim\sim\sim}^{(i)} \notin Adm_i(\Sigma)$. By (d) of the Lemma, this implies the existence of a $Q(\Sigma'')\ T_i' \underset{\Sigma}{\sim\sim\sim\sim}^{(i)} \in Adm_i(\Sigma)$, ruling out $P_{i-1}(\Sigma')\ T_i \underset{\Sigma}{\sim\sim\sim\sim}^{(i)}$. Now, clearly, T_i and T_i' are two different transitions, since otherwise $Q(\Sigma'')$ alone would already rule out $P_{i-1}(\Sigma')$. Since $Q(\Sigma'')\ T_i' \underset{\Sigma}{\sim\sim\sim\sim}^{(i)}$ rules out $P_{i-1}(\Sigma')\ T_i \underset{\Sigma}{\sim\sim\sim\sim}^{(i)}$, T_i' must be a correct transition, according to the definition of "ruling out". Now, again applying (c) of the Lemma and the hypothesis, we infer the existence of a $P_{i-1}(\Sigma'') \in Adm_{i-1}(\Sigma'')$. Using (e) of the Lemma, we can construct $P_{i-1}(\Sigma'')\ T_i' \underset{\Sigma}{\sim\sim\sim\sim}^{(i)} \in Adm_i(\Sigma)$.

By ensuring the admissibility of the interpretation obtained, we have, at the same time, lost the guarantee that the result will be optimal. This loss, however, does not worry us: sub-optimality can be shown to occur only as a consequence of having a non-vanishing error value for bold symbols within item sequences; setting these values equal to zero restores the guaranteed optimality. Since the error value in question has been chosen small compared to the other error values, a sub-optimal result, if at all, can only be so by a relatively small amount.

2.5.5. Implementation

In view of the complexity of the correctness proof for our method for ensuring admissibility, the actual implementation of the algorithm turns out surprisingly simple. In order to be able to test the candidates against their companion interpretations, it is unnecessary to keep the $Com_{\Sigma^*}(P_i(\Sigma))$, which would be cumbersome, or even information allowing their reconstruction. Instead, it suffices to remember, for each pair (Σ^*,Σ'), whether $com_{\Sigma^*}(P_{i-1}(\Sigma'))$ is defined, or, for short, whether $def_{i-1}(\Sigma^*,\Sigma')$ holds. Then, $P_{i-1}(\Sigma')\ T_i \underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}} \in Adm_i(\Sigma)$ if and only if T_i is a correct transition or $def_{i-1}(\Sigma^*,\Sigma')$ does not hold, where Σ^* is the state, if any, such that T_i^* is correct in the context of $\underset{\Sigma^*}{\overset{(i-1)}{\sim\sim\sim\sim}} T_i^* \underset{\Sigma}{\overset{(i)}{\sim\sim\sim\sim}}$. This follows immediately from the Companion Theorem and the definition of Com. Since such a Σ^* must differ from Σ', the diagonal values $def_{i-1}(\Sigma,\Sigma)$ are irrelevant.

The algorithm makes use of a number of variables:

[1:8] int ev. This will be used to accumulate the error values for the partial interpretations $P_i(\Sigma)$, for each of the eight states Σ, coded as an integer from 1 to 8. A value of *max int* is used to indicate the absence of a partial interpretation for a state.

[1:n] struct (int st sw, bool mark) trans. This variable is used to store the state switchers encountered at each of the n transitions. The *mark* field will indicate, at the completion of the algorithm, whether the corresponding state switcher is marked. At an intermediate stage it will indicate whether the partial interpretation $P_i(N)$ is a continuation of $P_{i-1}(\Sigma_C)$ (where C is the i-th state switcher), in which case the field is *false*, or of $P_{i-1}(N)$, in which case the field is *true*. In this way, *trans* contains sufficient information to follow the optimal interpretation back from the final state neutral. This is done at the last stage of the algorithm and at such places only where a transition $N \to N$ is chosen, the value *true* is retained for the *mark* field.

[1:8, 1:8] bool def. During the treatment of the i-th state switcher, *def*$[\Sigma^*,\Sigma']$ contains the value of $def_{i-1}(\Sigma^*,\Sigma')$. Initially, $com_{\Sigma^*}(P_0(\Sigma'))$ is

undefined for all $\Sigma^* \neq \Sigma'$, so that the entries of *def* are initially set to *false*. After the $P_i(\Sigma)$ have been determined (, which is trivial, except for $\Sigma = N$), *def* has to be updated. Let $P_i(\Sigma)$ be equal to $P_{i-1}(\Sigma')\ T_i\ \underset{\Sigma}{\overset{(i)}{\sim\sim\sim}}$. For $\Sigma^* \neq \Sigma$, $def_i(\Sigma^*,\Sigma) \Leftrightarrow com_{\Sigma^*}(P_i(\Sigma))$ is defined $\Leftrightarrow Com_{\Sigma^*}(P_i(\Sigma))$ is defined $\Leftrightarrow$ there exists a Σ^{**} such that T_i^* is a correct transition in the context of $\underset{\Sigma^{**}}{\overset{(i-1)}{\sim\sim\sim}}\ T_i^*\ \underset{\Sigma^*}{\overset{(i)}{\sim\sim\sim}}$ and $com_{\Sigma^{**}}(P_{i-1}(\Sigma'))$ is defined.

Case A: There does not exist such a correct predecessor Σ^{**} of Σ^*. In this case, $def_i(\Sigma^*,\Sigma)$ is false (for all Σ).

Case B: There exists a correct predecessor Σ^{**} and $\Sigma^{**} \neq \Sigma'$. Then $def_{i-1}(\Sigma^{**},\Sigma') \Leftrightarrow com_{\Sigma^{**}}(P_{i-1}(\Sigma'))$ is defined, so $def_i(\Sigma^*,\Sigma) = def_{i-1}(\Sigma^{**},\Sigma')$. Since in many cases the correct predecessor of Σ^{**} is Σ^*, and the actual predecessor of Σ' is Σ, this can often be achieved by swapping in *def* the Σ^* column with the Σ^{**} column and the Σ row with the Σ' row.

Case C: There exists a correct predecessor Σ^{**} and $\Sigma^{**} = \Sigma'$. This case may only arise at a fork in the interpretation paths, so $\Sigma^{**} = \Sigma' = N$. Since $com_N(P_{i-1}(N)) = P_{i-1}(N)$, which is always defined, $def_i(\Sigma^*,\Sigma)$ = true.

The algorithm is given in an ALGOL-68-like notation:

```
[ ] bool rowfalse = (false, false, false, false, false, false, false, false);

for S to 8
do ev[S]:= max int; def[S,]:= rowfalse od;
ev[N]:= 0;


for i to n
do    if bold symbol encountered
      then for S to 8
           do (S = N | ∿ |: ev[S] ≠ max int | ev[S] +:= error value bold
                ¢ see 2.4.3. ¢)
           od
```

```
        fi;
        int C = next state switcher; trans[i]:= (C, false);
        int ev i = error value type i,
        int ev ii = error value type ii,
        int ev iv = error value type iv ¢ see 2.4. ¢;
        int erval = ev[N] + ev i;
        if  ev ii = 0 ∧ def[Σ_C,N]
        then ev[N]:= ev[Σ_C]; ev[Σ_C]:= erval; swap (def[,Σ_C], def[,N]);
             if ev i = 0
             then swap (def[Σ_C,], def[N,])
             else def[N,]:= def[Σ_C,]; def[Σ_C,]:= rowfalse
             fi
      else int erval 1 = (ev[Σ_C] = max int | max int | ev[Σ_C] + ev ii),
             int erval 2 = ev[N] + ev iv;
             if  erval 2 > erval 1
             then  ev[N]:= erval 1; ev[Σ_C]:= erval; swap (def[,Σ_C], def[,N]);
                   if ev i = 0
                   then swap (def[Σ_C,], def[N,])
                   else def[N,]:= def[Σ_C,]; def[Σ_C,]:= rowfalse
                   fi
             else  ev[N]:= erval 2; ev[Σ_C]:= erval; mark of trans[i]:= true;
                   def[,Σ_C]:= def[,N];
                   if ev i = 0
                   then def[Σ_C,]:= def[N,]; def[Σ_C,N]:= true
                   else def[Σ_C,]:= rowfalse
                   fi;
                   def[N,]:= rowfalse
             fi
        fi
od;

int S:= N;
for i from n by -1 to 1
do    if S = N
```

```
        then (¬ mark of trans[i] | S:= Σ st sw of trans[i])
        else mark of trans[i]:= false;
             (S = Σ st sw of trans[i] | S:= N)
        fi
od
```

2.6. An example

The program given below is example 11.12 from [2]. It was punched by a punch card operator who was instructed not to correct any errors (so these may be considered typical). As for the state switchers, two errors have been made:

(i) In line 10, which ought to run: *name."l,* a quote-symbol has been replaced by an equals-symbol. The algorithm marked the quote-symbol on line 11 (pointed at by an arrow). Since the first quote-symbol on that line cannot be marked, for, otherwise, an inadmissible transition $\Sigma_{''} \overset{''}{\rightarrow} \Sigma_{''}$ would result, this is the best possible correction.

(ii) In line 14, which ought to start with *"for.a.listing,* a quote-symbol has been inserted. The algorithm marked the next quote-symbol. One might wonder why not the first quote-symbol on that line was marked. Indeed, the error values for both interpretations are equal. In such a case, the interpretation ending in a transition $N \overset{C}{\rightarrow} N$ prevails, the rationale being that thereby in most cases the interpretation is chosen which, from left to right, is the last one to have an incorrect transition.

```
begin
  mode ra = ref auth, rb = ref bock;
  mode auth = struct(string name, ra next, rb book),
    book = struct(string title, rb next);
  ra auth, first auth := nil, last auth;
  rb book; string name, title; int i; file input, output;
  open (input, "", remote in); open (output, "", remote out);
  putf (output, ($p
    "to.enter.a.new.author.""author"",.a.space,.and.his.
(10) name.=1
    "to.enter.a.new.book,.type.""book"",.a.space,.the.name.of.
    the.author,.a.new.line,.and.the.title."1
(14) "for".a.listing.of.the.books.by.an.author,.type.""list"",
    .a.space,.and.his.name."|
    "to.find.the.author.of.a.book,.type.""find"",.a.new.line,
    .and.the.title."1
    "to.end,.type.""end""""al$, ".."));
proc update = void :
  if ra (first auth) :=: nil
  then auth := first auth := last auth := heap auth :=
    (name, nil, nil)
  else auth := first auth;
    while ra (auth) : ≠: nil
    do
      (name = name of auth | goto known | auth := next of auth)
    od;
    last auth := next of last auth := auth := heap auth :=
      (name, nil, nil);
known: skip
  fi;
do
try again:
  getf (input, ($c("author","book","list","find","end",""),
      x30al, 80al$, i));
  case i in
    # author # (getf (input, name): update),
    # book #
    begin getf (input, (name, title)); update;
      if rb (book of auth) :=: nil
        then book of auth := heap book := (title, nil)
        else book := book of auth;
          while rb (next of book) :≠: nil
            (title = title of book | goto try again
            | book := next of book)
          od;
          (title ≠ title of book | next of book := heap book :=
            (title, nil))
      fi
    end,
```

```
      # list #
      begin getf (input, name); update;
        putf (output, ($p"author:."30all$, name));
        if rb (book := book of auth) :=: nil
        then put (output, ("nc.publications", newline))
        else on page end (output,
          (ref file f) bool :
            (putf (f, ($p"author:."30a41k"continued"11$, name));
            true));
          while re (book) :≠: nil
          do
            putf (output, ($180a$, title of book));
              book := next of book
          od;
          on page end (output, (ref file f) bool : false)
        fl
      end
      # find #
      begin getf (input, (loc string, title)); auth := first auth;
        while ra (auth) :≠: nil
        do book := book of auth;
          while rb (book) :≠: nil
          do
            if title = title of book
            then putf (output, ($1"author:."30a$, name of auth));
              goto try again
              else book := next of book
              fl od;
              auth := next of auth
        od;
        put (output, (newline, "unknown", newline))
      end
      # end # (put (output, (new page, "signed.off", close));
        close (input); goto stop),
        # error # (put (output, (newline, "mistake,.try.again"));
          newline (input))
      esac
    od
  end
```

References

[1] L.G.L.T. Meertens and J.C. van Vliet, *Repairing the parenthesis skeleton of ALGOL 68 programs*, Report IW 2/73, Mathematical Centre, Amsterdam, February 1973.

[2] A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.G.L.T. Meertens and R.G. Fisker, *Revised Report on the Algorithmic Language ALGOL 68*, to be published.

[3] D.V. Lindley, *Introduction to Probability and Statistics*, Vol. I, II, Cambridge University Press, 1965.

[4] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research*, Holden-Day, Inc., San Francisco, 1969.

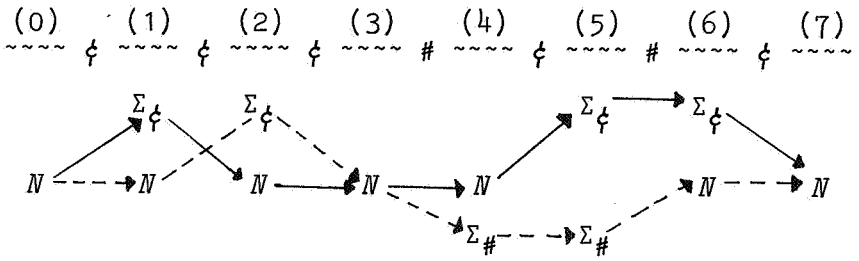


Fig. 1. Two locally admissible interpretations

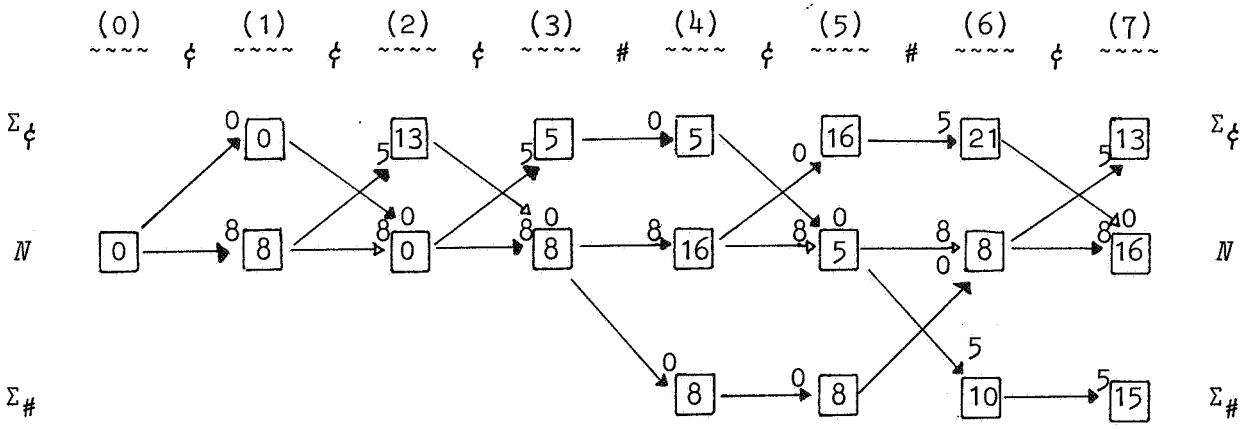


Fig. 2. Obtaining an optimal interpretation

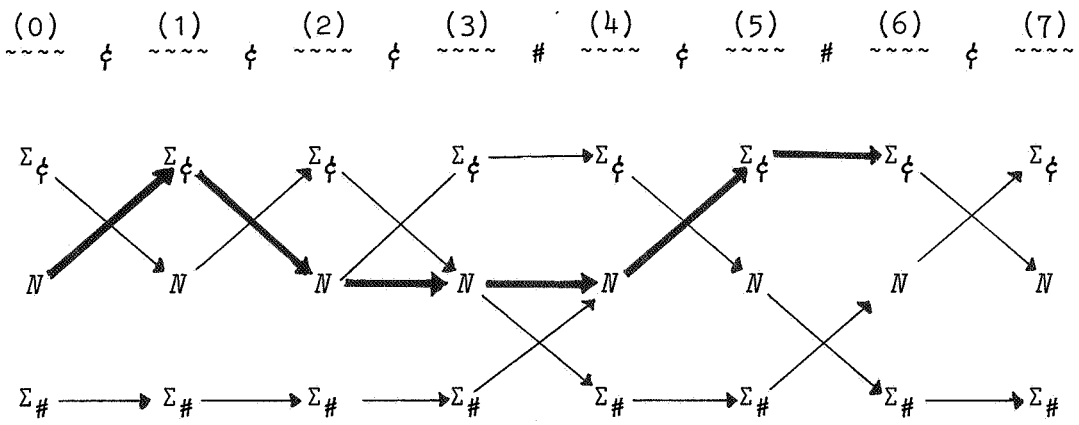


Fig. 3. The interpretation $P_6(\Sigma_{\text{¢}})$.

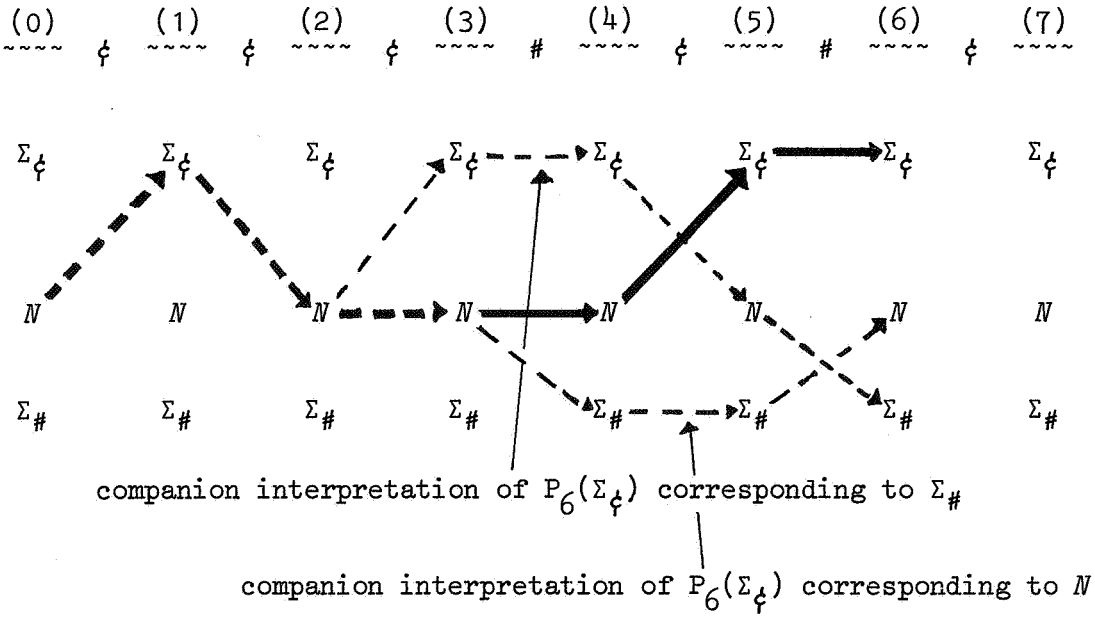


Fig. 4. Two companion interpretations of $P_6(\Sigma_¢)$.