SEMIPAL 2, an extension of the mathematical notational language SEMIPAL

de Bruijn, N.G.

Published: 01/01/1969

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

Take down policy
If you believe that this document breaches copyright please contact us (openaccess@tue.nl) providing details. We will immediately remove access to the work pending the investigation of your claim.

Download date: 24. Jan. 2019
SEMIPAL 2, an extension of the mathematical notational language SEMIPAL.

by N.G. de Bruijn.

1. SEMIPAL is a simple language, or rather a notational system for the abbreviation of expressions. It was described in [2], sec. 4.3-4.8. In this note we shall extend that language by extending the notion of the indicator strings used in SEMIPAL.

In SEMIPAL every line represents the definition of a new identifier in terms of previously introduced ones, as far as the new identifier was not introduced as a variable or as a primitive notion. And every line has a context indicator, which is either 0 or a previous block opener. If the indicator of a line is $x_n$, and if, for every $j (1 \leq j \leq n)$, $x_{j-1}$ is the indicator of the line whose identifier is $x_j$, and if 0 is the indicator of the line whose identifier is $x_1$, then $(x_1, \ldots, x_n)$ is called the indicator string of the given line.

With this definition of the indicator strings, the following property holds. If $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$ are the indicator strings of two different lines, then there is an integer $k (0 \leq k \leq \min(n, m))$ such that $x_i = y_i$ for all $i = 1, \ldots, k$, and such that there is no further case where an $x_i$ equals an $y_j$. In the extended form of SEMIPAL, to be discussed presently, this property no longer holds, and the indicator strings can no longer be obtained from single indicators.

In SEMIPAL 2, as we shall call this version, we shall write an indicator string in front of every line, and no longer a single indicator.

2. In order to facilitate our discussion, we number the lines by integers $1, 2, 3, \ldots$; the $m$-th line is called $\lambda_m$, the indicator string of $\lambda_m$ is $\mu_m$; the identifier part of $\lambda_m$ is $a_m$, the definition part of $\lambda_m$ is $b_m$. The order in the indicator strings is, as is usually done in strings, referred to as
left to right, with left < right.

The following rules define SEMIPAL 2 uniquely.

(i) \( \delta_m \) is either the symbol \(-\), or the symbol PN, or an expression (see sec. 3).

(ii) \( a_m \) is called a block opener if \( \delta_k = - \).

(iii) \( \mu_m \) is either 0 or a linearly ordered set of block openers taken from the sequence \( (a_1, \ldots, a_{m-1}) \). The order in \( \mu_m \) need not be the same as in \( (a_1, \ldots, a_{m-1}) \).

(iv) If \( a_j \) occurs in \( \mu_m \), then \( \mu_j \) is a subset of \( \mu_m \). Moreover, the order in \( \mu_j \cup \{a_j\} \) is preserved in \( \mu_m \) (the order in \( \mu_j \cup \{a_j\} \) is defined by taking the existing order in \( \mu_j \), and requiring \( b < a_j \) for all \( b \in \mu_j \)).

(v) If \( \delta_m \) is an expression, then the pair \( (m, \delta_m) \) is a "valid pair" (see sec. 3).

3.1 The notion "expression" (or "parentheses expression") is defined by recursion: if \( p \) is an identifier then \( p \) is an expression; if \( p \) is an identifier and if \( \Sigma_1, \ldots, \Sigma_k \) are expressions, then \( p(\Sigma_1, \ldots, \Sigma_k) \) is an expression.

3.2 We shall also define the notion of "valid pair" by recursion. The letters \( m, j, n, k \) will stand for integers.

(i) If \( a_j \in \mu_m \) (whence \( a_j \) is a block opener), then \( (m, a_j) \) is a valid pair.

(ii) Assume that \( 1 \leq j < m \), and that \( a_j \) is not a block opener. Let \( n \) be the number of elements of \( \mu_j \), let \( k \) be an integer satisfying \( 0 \leq k < n \). Let \( \Sigma_1, \ldots, \Sigma_k \) be such that \( (m, \Sigma_1), \ldots, (m, \Sigma_k) \) are valid pairs. And assume that the first \( n-k \) entries of \( \mu_j \) occur in \( \mu_m \). Under these circumstances \( (m, \Sigma) \) is also a valid pair, where

\[ \Sigma = a_j(\Sigma_1, \ldots, \Sigma_k). \]

These descriptions are to be modified in the obvious way if \( k = n \), or \( k = 0 \), or both.

4. Examples. The SEMIPAL book in [2], sec. 4.1 (categories to be omitted) can be transformed at once into the following SEMIPAL 2 book:
Remark 1.

In some cases (see \( f := \text{PN} \)) we define a thing in terms of what seems to be independent variables, in other cases (like \( q := f(w) \)) the variables seem to be less independent. There does not seem to be much of a reason to admit the latter possibility, but it will become significant if we wish to attach a category to each variable, like we do in PAL. In that case, the category of a variable \( w \) may depend on the variables occurring in the indicator string of the line where \( w \) is introduced.

Remark 2.

It may help the reader to write the identifiers (apart from the block openers) as functions of the variables in the indicator string:
Remark 3.

Note that the variable $w$ (see above) has $x$ in its indicator string, and that this has the effect that $x$ has to occur in every indicator string containing $w$.

5. Completion.

If $(k, \Sigma)$ is a valid pair, then we define the "completion" $\Sigma'$ of $\Sigma$. (cf. [2], see 4.6).

(1) If $\Sigma$ is a single block opener, then $\Sigma' = \Sigma$
(2) Let $\Sigma = a_j(\Sigma_1, \ldots, \Sigma_k)$ (for notation see sec. 3.1), and let $b_1, \ldots, b_{n-k}$ be the first $n-k$ entries of $\mu_j$. Then

$$\Sigma' = a_j(b_1, \ldots, b_{n-k}, \Sigma_1, \ldots, \Sigma_k),$$

with obvious modifications if $k = n$, or $k = 0$, or both.


Let $(k, \Sigma)$ be a valid pair. We shall give a recursive definition of the "normal form" of $\Sigma$. It will become an expression $\Sigma^*$ that again has the property that $(k, \Sigma^*)$ is a valid pair.

(i) If $\Sigma$ is a single block opener then $\Sigma^* = \Sigma$.
(ii) If $\Sigma$ is not a single block opener, we first form the completion $\Sigma'$ of $\Sigma$. Let

$$\Sigma' = a_j(\Sigma_1, \ldots, \Sigma_n)$$

(possibly with $n = 0$).
If $\delta_j = \text{PN}$, we define $\Sigma^*$ by
If \( \delta_j \) is an expression, then we first produce the normal form \( \delta_j^* \) of \( \delta_j \). Let \( \mu_j = (u_1, \ldots, u_n) \) be the indicator string of the \( j \)-th line. The expression \( \delta_j^* \) may contain \( u_1 \) at various places, it may contain \( u_2 \) at various places, etc. We now replace in \( \delta_j^* \) each of these \( u_1 \)'s by \( \Sigma_1 \), each of the \( u_2 \)'s by \( \Sigma_2 \), etc. (This substitution refers only to the \( u_1 \)'s, \( u_2 \)'s, \ldots that were originally present in \( \delta_j^* \), and not to those \( u_i \)'s that enter into the formula since they occurred in \( \Sigma_1, \Sigma_2, \ldots \).) This substitution procedure leads to what we call \( \Sigma^* \).

**Remark.**

If both \((k, \Sigma)\) and \((m, \Sigma)\) are valid, then the normal form evaluated with reference to \( k \) is identical to the one evaluated with reference to \( m \).

7. Every SEMIPAL book \( \Lambda \) turns into a SEMIPAL 2 book \( \Lambda' \) if we attach to every line an indicator string obtained from the indicators in the way described in sec. 1. The normal form of an expression in \( \Lambda \) (normal form as defined for SEMIPAL in [2, sec. 4.7]) will be identical to the normal form of the same expression as defined in SEMIPAL 2 by means of sec. 6 above.

8. Let \( \Lambda \) be a correct SEMIPAL 2 book. An expression \( \Sigma \) is called a \( \Lambda \)-expression if there is a \( k \) such that \((k, \Sigma)\) is a valid pair (with respect to \( \Lambda \)).

Every \( \Lambda \)-expression has a uniquely defined normal form. Two \( \Lambda \)-expressions are called definitionally equivalent if they have the same normal form.

In order to test definitional equivalence it is not always necessary to evaluate the normal forms. One of the devices that will enable us to reduce the amount of work is the "reduced indicator string", to be discussed presently.

9. **Reduced indicator strings.**

If \( \delta_j \) is an expression, we define the reduced indicator string \( \widetilde{\mu}_j \) as the string obtained from \( \mu_j \) by deleting all variables that do not occur in the normal form of \( \delta_j \).

A correct book can sometimes be made incorrect if we replace all \( \mu_j \)'s by \( \widetilde{\mu}_j \)'s. In the first place we might have to omit subexpressions whose position in an expression refers to a variable omitted from the corresponding indicator string. Secondly we should bear in mind that replacing
\( \mu_m \) by \( \tilde{\mu}_m \) might endanger the validity of rule (iv) of sec. 2.

10. We can extend PAL and AUTOMATH (described in [1], [2]) to PAL 2 and AUTOMATH 2 by means of the same rules on indicator strings as discussed for the case of SEMIPAL in this note.

AUTOMATH 2 has some obvious advantages over AUTOMATH. It can be easier to write, and may require a smaller number of lines. On the other hand, disadvantages of AUTOMATH 2 are:

(i) Writing indicator strings (and reduced indicator strings) can give more trouble than writing single indicators.

(ii) We lose the easy structure of a book, as illustrated by the "tree of knowledge".

Even if we write AUTOMATH instead of AUTOMATH 2, it may be very useful to evaluate the reduced indicator string for every line. This can of course be done by an AUTOMATH processor. By means of these reduced indicator strings, the processor can economize considerably on the amount of checking it has to do.

References:
