Normal forms for a class of formulas

Citation for published version (APA):

Document status and date:
Published: 01/01/1987

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 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

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.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Normal forms for a class of formulas
by
H.M.M. ten Eikelder
J.C.F. Wilmont
87/16
december 1987
This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science of Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.
NORMAL FORMS FOR A CLASS OF FORMULAS

H.M.M. TEN EIKELDER
J.C.F. WILMONT

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
Eindhoven, The Netherlands

Abstract:

A class of formulas which consist of real functions $a_1, \ldots, a_N$, their derivatives and integration operators $I$ is considered. Formulas of this type arise in some parts of mathematical physics. Due to partial integration, various formulas can have the same meaning. A normal form and a normalizing algorithm are given.
1. INTRODUCTION

Formula manipulation techniques are used nowadays in various parts of science. In this paper, we shall discuss a formula manipulation problem which arises in a part of mathematical physics. In that field the work on partial differential equations considered as Hamiltonian systems has evolved rapidly the last decennium. The verification of several properties of a class of these equations (in particular computations which are related to the recursion operator and its Nijenhuis tensor) leads to the class of formulas considered in this paper.

Loosely speaking, these formulas consist of polynomials in smooth functions $a_1, \ldots, a_N: \mathbb{R} \rightarrow \mathbb{R}$ and their derivatives, and integration operators $\mathcal{I}$. Different expressions of this type can have the same meaning. For instance, if differentiation is denoted by a subscript $x$, the expressions $\mathcal{I}(a_1 a_2) + \mathcal{I}(a_1 a_2)$ and $a_1 a_2$ have the same meaning (under appropriate boundary conditions and definition of $\mathcal{I}$). This means that to verify if some sum of formulas vanishes, it is not sufficient to see if the coefficients of all appearing formulas cancel out. The problem can be solved by introducing normal forms for the considered type of formulas. Then a sum of different formulas in normal form should only vanish if all the coefficients vanish. In this paper such a normal form is given. We also describe an algorithm that transforms a formula to its normal form. Explicit examples of the computation mentioned above can be found in for instance Ten Eikelder (1986) or Fuchssteiner et al. (1987). The latter paper also gives some heuristic considerations on normal forms. However, the normal form and normalizing algorithm presented in this paper are not given.

The organization of this paper is as follows. In Section 2 we give the syntax and semantics of the considered class of formulas. Some introductory contemplations on the problem of finding a normal form will be given in Section 3. We shall formulate a hypothesis which is a sufficient condition for constructing normal forms. In Sections 4 and 5 we assume that this hypothesis holds. In Section 4 we describe the class of formulas in normal form and give a normalizing algorithm. The property that two formulas in normal form have the same meaning (semantics) if and only if they are equal (syntax) is proved in Section 5. Then, in Section 6 we return to
the hypothesis and show that it can be satisfied. Finally, some concluding remarks are given in Section 7.

2. THE CLASS OF FORMULAS

Let $T$ be a set of syntactic representations of monomials in $a_1, \ldots, a_N$ and their (higher) derivatives. We shall adopt the usual notation in mathematical analysis to write these monomials, i.e. elements of $T$ are, for instance, the following 'strings':

$$1 \text{ (empty product), } a_1 a_2 a_3^2 x, \ a_1 a_4^3 a_2 x,$$

Elements of $T$ will be called terms. The set of formulas $F$ is generated by the following grammar:

$$f ::= t \quad (t \in T)$$
$$f ::= I(f) \quad \text{(not } I(1))$$
$$f ::= ff.$$

So, $F$ consists of all well-formed expressions which can be constructed using terms and the symbols $I$, $( $ and $)$, except expressions which contain $I(1)$. Elements of $F$ are, for instance,

$$a_1 a_4^3 a_2 x, \ a_1 a_2 I(a_1 x I(a_2^2 x) I(I(a_2 x a_3))).$$

The set of sumformulas $SF$ is defined by

$$SF = \left\{ \sum_{i=1}^{m} \lambda_i f_i \mid m \geq 0, (\forall i: 1 \leq i \leq m: \lambda_i \in \mathbb{Q}, f_i \in F) \right\},$$

where the metasymbol $\sum$ has the usual meaning. So, a sumformula is a sum of formulas with rational coefficients, for instance

$$I(a_1 x I(a_2^2 a_2 x) I(a_2 x a_3)) - \frac{1}{3} a_1 a_2 a_3^3 + \frac{1}{2} a_1 a_2 a_3^3 I(a_1 x a_2 a_3 x) +$$
$$+ \frac{1}{6} a_1 a_2^2 a_3 I(a_2 a_3 x) - \frac{1}{6} I(a_1 a_2 a_3^3 x) + \frac{1}{6} I(a_1 a_2 a_3^3 x). \quad (2.1)$$
Of course, more formal syntactic definitions of terms, formulas and sumformulas can be given. However, for our purpose the informal description given here is sufficient.

Next, we describe the semantics or meaning of terms, formulas and sumformulas. Let $C$ be a set of infinitely differentiable functions $\mathbb{R} \to \mathbb{R}$, which together with their derivatives vanish sufficiently fast if the independent variable $x \to -\infty$. The precise structure of $C$ is not important here. If $a_1, \ldots, a_N \in C$ and $I(h)(x) = \int_{-\infty}^{\infty} h(y) dy$, then an element of $T, F$ or $SF$ can be considered as a function $\mathbb{R} \to \mathbb{R}$, written in the usual notation in mathematical analysis. So, the semantics of an element $T, F$ or $SF$ is a mapping $C \to C$, where $\hat{C}$ is also a set of functions $\mathbb{R} \to \mathbb{R}$. (Since $1 \in T$, the set $\hat{C}$ must contain all constant functions.) Clearly different terms or (sum)formulas can have the same meaning. For instance, $\int \frac{1}{3} a_1 a_1 x I(a_2)$ has the same meaning as $\frac{2}{3} a_1 I(a_2) a_1 x - \frac{1}{3} I(a_2) a_1 a_1 x$. From now on, we shall identify all formulas which can be transformed into each other by the usual algebraic operations (i.e. interchanging elements of terms or formulas, interchanging formulas in a sumformula, summing coefficients of identical formulas, etc.). So, every term or (sum)formula represents in fact an equivalence class of terms or (sum)formulas and $g_1 = g_2$ means that $g_1$ and $g_2$ belong to the same equivalence class. By introducing an ordering on $T, F$ and $SF$, it is always possible to compute a unique representative for each equivalence class. We shall always assume that, if $\sum_{i=1}^{m} \lambda_i f_i$ is a (representative of a class of) sumformula(s), the number $m$ is as small as possible. This is equivalent to saying that the coefficients $\lambda_i$ do not vanish and that $f_i \neq f_j$ for $i \neq j$.

If two (sum)formulas $g_1$ and $g_2$ have the same meaning we shall write $g_1 \equiv g_2$. Clearly, $\equiv$ is an equivalence relation. For instance, $I(a_1 x) \equiv a_1$, but $I(a_1 x) \neq a_1$. More complicated different sumformulas with the same meaning can easily be found using partial integration.

Let $\mathcal{V}: SF \to SF$ be the 'syntactic differential operator'. A formal inductive definition of $\mathcal{V}$ can easily be given ($\mathcal{V}(fI(g)) = fg + \mathcal{V}(f)I(g)$, etc.), but we shall not do that here. The well-known partial integration formula from mathematical analysis now yields

$$I(f \mathcal{V}(g)) \equiv fg - I(g \mathcal{V}(f)),$$  \hfill (2.2)
or equivalently
\[ I(f)I(g) \equiv I(fI(g)) + I(gI(f)). \]  
(2.3)

An elementary computation using these relations shows that (2.1) has the same meaning as \( O \). Hence, there exist different sumformulas which have the same meaning. This raises the need for a normal form for sumformulas. In the sequel we shall describe a subset \( S_N \) of the set of sumformulas \( S_F \) such that

i) every sumformula in \( S_F \) can be transformed to a sumformula in \( S_N \) with the same meaning,

ii) two sumformulas in \( S_N \) have the same meaning if and only if they are equal.

Algebraically \( S_N \) is isomorphic with \( S_F / \equiv \), but, since we do not yet have an algorithm that verifies if two sumformulas have the same meaning, this observation is not of much practical use.

Finally, we introduce some additional notations and conventions. The set of sumterms \( S_T \) is defined by

\[ S_T = \left\{ \sum_{i=1}^{m} \lambda_i t_i \mid m \geq 0, (\forall i : 1 \leq i \leq m : \lambda_i \in Q, t_i \in T) \right\}. \]

Then, \( T \subseteq S_T \subseteq S_F \) (also \( T \subseteq F \subseteq S_F \)). In the sequel we shall also be a little less formal in the notation, for instance, if \( ff = \sum_{i=1}^{m} \lambda_i f_i \) is a sumformula, then \( I(ff) \) stands for \( \sum_{i=1}^{m} \lambda_i I(f_i) \), etc. For the types of variables we always use the following conventions:

\[ t, t_1, t_2, \ldots, s, s_1, s_2, \ldots \in T, \]

\[ tt, tt_1, tt_2, \ldots, ss, ss_1, ss_2, \ldots, uu \in S_T, \]

\[ f, f_1, f_2, \ldots \in F, \]

\[ ff \in S_F, \]

\[ \lambda, \lambda_1, \lambda_2, \ldots \in Q. \]
3. NECESSARY CONDITIONS FOR NORMAL FORMS

We first study normal forms for sumterms and for sumformulas of the form \( I(tt) \). The following elementary theorem shows that sumterms can be considered as being in normal forms.

Theorem 3.1:
For all \( tt \in ST : tt = 0 \iff tt \equiv 0 \).

Proof:
Any sumterm can be considered as a polynomial in a number of variables which is a finite subset of \( \{a_1, a_{1x}, a_{xx}, \ldots, a_N, a_{Nx}, a_{xx} \} \). Moreover, every set of values for these variables can be obtained as the corresponding derivatives of functions \( a_1, a_2, \ldots, a_N \in C \) in an arbitrary point \( x \in \mathbb{R} \).

The theorem now follows from the standard result in algebra that a polynomial that vanishes for all values of its arguments is the zero polynomial, see for instance Lang (1965).

An equivalent formulation of this theorem is that two sumterms are equal if and only if they have the same meaning.

Next, consider normal forms for a sumformula of the form \( I(tt) \). A simple computation shows that

\[
I(a_1 a_2 + a_{1x} a_{xx}) = a_1 a_2 - a_{1x} a_{xx} + 2I(a_{xx} a_{xx}).
\]

This suggests to try \( tt_1 + I(tt_2) \) as normal form for \( I(tt) \), where the sum-terms \( tt_1 \) and \( tt_2 \) possibly must satisfy additional conditions. In particular, \( tt_2 \) is intended to contain terms which cannot be 'integrated further' in some way. Let \( ss_1 + I(ss_2) \) also be a 'normal form' for \( tt \), then from

\[
(3.1) \quad tt_1 = ss_1 \iff I(ss_2 - tt_2)
\]

we must be able to conclude that \( tt_1 = ss_1 \) and \( tt_2 = ss_2 \). From (3.1) and Theorem 3.1 we see that \( ss_2 = tt_2 \) implies \( ss_1 = tt_1 \). So, it is sufficient to find additional conditions such that (3.1) implies \( ss_2 = tt_2 \). This can be obtained in the following way. Suppose \( NIT \) (nonintegrable (sum)term) is a predicate on \( ST \) such that
A.\[ (\forall i: 1 \leq i \leq m: \text{NIT}(t_i)) \]

and for \( uu \neq 0 \)

\[ \text{NIT}(uu) \Rightarrow (\forall ss: ss \in \text{ST}: I(uu) \neq ss). \] (3.3)

So, if \( \text{NIT}(uu) \) holds and \( uu \neq 0 \), then \( uu \) cannot be the derivative of a sum-term. Clearly, if in (3.1) \( \text{NIT}(ss_2) \) and \( \text{NIT}(tt_2) \) hold, then also \( \text{NIT}(ss_2 - tt_2) \) and (3.3) yields \( ss_2 = tt_2 \). So \( tt_1 + I(tt_2) \) can be considered as a normal form of \( I(tt) \) if \( \text{NIT}(tt_2) \) holds. In Sections 4 and 5, we shall assume that it is always possible to construct this type of 'normal form' for \( I(tt) \). Formally, in Sections 4 and 5 we assume the Hypothesis H:

There exists a predicate \( \text{NIT} \) on \( \text{ST} \) that satisfies (3.2) and (3.3) and there exist mappings \( \text{Int}: \text{ST} \rightarrow \text{ST} \) and \( \text{Rest}: \text{ST} \rightarrow \text{ST} \) such that

\[ I(tt) \equiv \text{Int}(tt) + I(\text{Rest}(tt)) \] (3.4)

and

\[ \text{NIT}(\text{Rest}(tt)). \] (3.5)

It turns out that, if this hypothesis holds, normal forms for sumformulas with an arbitrary number of I's can easily be constructed.

In Section 6 we shall construct a predicate \( \text{NIT} \) and mappings \( \text{Int} \) and \( \text{Rest} \) which satisfy the hypothesis H. Note that \( D, \text{Int}, \text{Rest} \) (and the mappings \( M_1, M_2 \) and \( M \) of Section 4) are mappings from sumformulas or sumterms to sumformulas or sumterms while I is a symbol which actually appears in (sum)formulas.

4. THE NORMALIZING ALGORITHM

We shall now describe a subset \( \text{SN} \) of the set of sumformulas \( \text{SF} \). The main result of this section is Theorem 4.1, which states that for every sum-formula in \( \text{SF} \) a sumformula in \( \text{SN} \) can be constructed which has the same meaning.
First we introduce basis formulas in normal form. For each \( k \in \mathbb{N} \) the set \( B_k \) of basis formulas in normal form with order \( k \) is recursively defined by

\[
B_0 = \{1\},
\]

\[
B_{k+1} = \{I(tb) \mid t \in T, b \in B_k, NIT(t)\}.
\]

The set of basis formulas \( B \) is then given by

\[ B = \bigcup_{k \geq 0} B_k. \]

The set \( N \) of formulas in normal form is defined by

\[ N = \{tb \mid t \in T, b \in B\}. \]

So, a formula in normal form consists of the product of a term and a basis formula. Clearly, \( T \subseteq N \subseteq F \). The order \( \sigma \) of a formula in \( N \) is defined by:

\[ \sigma(tb) = k \text{ if } b \in B_k. \]

So, \( \sigma(n) \) is nothing but the number of I's in \( n \in N \). A formula \( n \in N \) with order \( k \) can be written as

\[ n = t_0 I(t_1) I(t_2) \ldots I(t_k) \ldots, \]

with \( t_i \in T \) for \( i = 0, \ldots, k \) and \( NIT(t_i) \) for \( i = 1, \ldots, k \).

Then \( ST \subseteq SN \subseteq SF \). We generalize the notion of order to \( SN \) by

\[
\sigma\left( \sum_{i=1}^{m} \lambda_i n_i \right) = \begin{cases} 0 & \text{if } m = 0, \\ \text{MAX} \{ \sigma(n_i) \} & \text{if } m \geq 1. \end{cases}
\]
By gathering formulas which have the same basis formula, every sumformula $nn \in SN$ can be written as

$$nn = \sum_{i=1}^{m} tt_i b_i , \quad tt_i \in ST, b_i \in B \text{ for } i = 1, \ldots, m ,$$

(4.1)

where the basis formulas $b_i$ are mutually different and $tt_i \neq 0$ for $i = 1, \ldots, m$. If in the sequel of this paper a sumformula $nn \in SN$ is written in the form (4.1), we shall always assume that these restrictions on the $tt_i$ and $b_i$ hold.

In addition to the convention given in Section 2, we agree that always

$$b_1, b_2, \ldots, c_1, c_2, \ldots, e_1, e_2, \ldots \in B ,$$

$$n_1, n_1, n_2, \ldots \in N ,$$

$$nn, nn_1, nn_2, \ldots \in SN .$$

In the remaining part of this section we construct a mapping $M : SF \rightarrow SN$, which maps every sumformula to its normal form.

Suppose $ttb$ is a sumformula in normal form. Then, since not necessarily $NIT(tt)$ holds, $I(ttb)$ may not be in normal form. We first describe a mapping which gives a normal form for $I(ttb)$. A simple calculation using the derivative of (3.4) and partial integration (2.2) yields

$$I(ttb) \overset{M}{=} \left( D(\text{Int}(tt) + \text{Rest}(tt))b \right)$$

$$\overset{M}{=} \text{Int}(tt)b - I(\text{Int}(tt)D(b)) + I(\text{Rest}(tt)b) .$$

(4.2)

The first and, since $NIT(\text{Rest}(tt))$ holds, the last expression in (4.2) consists of formulas in normal form. If $b \in B_0$, then $D(b) = D(1) = 0$ and (4.2) yields a normal form for $I(tt)$. If $b \in B_k$ with $k \geq 1$, a normal form for $I(ttb)$ can be computed from (4.2) if a normal form for $I(\text{Int}(tt)D(b))$ is known. Since $O(\text{Int}(tt)D(b)) = k - 1$ and $O(ttb) = k$, we can use recursion to compute the normal form of $I(ttb)$. Define $M : SN \rightarrow SN$ by
The proof of the following lemma is now almost trivial.

**Lemma 4.1:**

For all \( nn \in SN \):

\[
M_I(nn) \equiv I(nn).
\]

So, if \( nn \) is in normal form, \( M_I(nn) \) is the normal form of \( I(nn) \).

Next, we discuss how a normal form of the product of two formulas in normal form can be computed. Consider two basis formulas. If (at least) one of them is element of \( B_0 \), then their product is trivially in normal form. Now consider the basis formulas \( I(tb) \) and \( I(sc) \). Partial integration (2.3) yields

\[
I(tb)I(sc) \equiv I(tbI(sc)) + I(scI(tb)).
\] **(4.4)**

Suppose that normal forms for the products of basis formulas \( bI(sc) \), respectively \( cI(tb) \) are known. Then, using the mapping \( M_I \), a normal form for the product \( I(tb)I(sc) \) can easily be computed from (4.4). Since

\[
\partial(b) + \partial(I(sc)) = \partial(c) + \partial(I(tb)) < \partial(I(tb)) + \partial(I(sc)),
\]

a normal form of the product of two basis functions can be computed recursively. Define \( M_2 : SN \times SN \rightarrow SN \) by

\[
M_2(1,c) = c,
\]

\[
M_2(b,1) = b,
\]

\[
M_2(I(tb),I(sc)) = M_I(tM_2(b,I(sc))) + M_I(sM_2(c,I(tb))),
\] **(4.5)**

\[
M_2(\sum \text{tt}_i \text{b}_i, \sum \text{ss}_j \text{c}_j) = \sum_i \sum_j \text{tt}_i \text{ss}_j \text{M}_2(\text{b}_i,\text{c}_j).
\]
Using induction with respect to the structure of \( nn1 \) and \( nn2 \) the following lemma can easily be proved.

**Lemma 4.2:**

For all \( nn1, nn2 \in SN \):

\[
M_2 (nn1, nn2) = nn1 \cdot nn2 .
\]

So, the mapping \( M_2 \) yields a normal form for the product of two sumformulas in normal form.

Using the mappings \( M_1 \) and \( M_2 \) it is easy to construct a mapping \( M: SF \rightarrow SN \) which transforms a sumformula from \( SF \) to its normal form. Recall that every formula \( f \in F \) is of the form \( t, I(f_1) \) or \( f_1 f_2 \) with \( t \in T, f_1, f_2 \in F \).

Define \( M: SF \rightarrow SN \) by

\[
M(t) = t ,
M(I(f)) = M_1(M(f)) ,
M(f_1 f_2) = M_2(M(f_1), M(f_2)) ,
M\left( \sum_{i=1}^{m} \lambda_i f_i \right) = \sum_{i=1}^{m} \lambda_i M(f_i) .
\]

Since every argument of \( M \) in a right-hand side of (4.6) is shorter than the corresponding argument in the left-hand side, this is a correct definition (i.e. \( M \) is defined by structure induction).

The main result of this section is the following

**Theorem 4.1:**

For all \( ff \in SF \):

\[
M(ff) = ff .
\]

**Proof:**

Using induction on the structure of \( ff \) and the Lemmas 4.1 and 4.2, the proof is almost trivial.
So, for every sumformula \( ff \) in \( SF \) a sumformula in \( SN \) with the same meaning is given by \( M(ff) \). Note that the definitions of the mappings \( M_1, M_2 \) and \( M \) are recursive; these mappings can easily be implemented by (recursive) functions.

5. UNIQUENESS OF NORMAL FORMS

In the preceding section we have described a subset \( SN \) of the set of sumformulas \( SF \). We have shown that for every sumformula \( ff \in SF \) a sumformula \( M(ff) \), the normal form of \( ff \), can be computed such that \( ff \) and \( M(ff) \) have the same meaning. It remains to be shown that \( M(ff) \) is the only element of \( SN \) which has the same meaning as \( ff \). That will be done in Theorem 5.1.

First, we introduce some notation and give three lemmas.

As explained in Section 4, every sumformula in normal form \( nn \) can be written as

\[
 nn = \sum_{i=1}^{m} t_i b_i, \tag{5.1}
\]

with \( m \) minimal. For each \( k \in \mathbb{N} \) the mapping \( \Pi_k : SN \rightarrow SN \) is defined in the following way. If \( nn \) is given by (5.1), then

\[
 \Pi_k(nn) = \sum_{\substack{i=1 \\ \text{such that} \ \ \theta(b_i) = k}}^{m} t_i b_i.
\]

So, \( \Pi_k(nn) \) is the sum of all formulas in \( nn \) with order \( k \) (if any). The width \( \mathcal{W}(nn) \) of \( nn \) given by (5.1) is defined as

\[
 \mathcal{W}(nn) = \sum_{\substack{i=1 \\ \text{such that} \ \ \theta(b_i) = \theta(nn)}}^{m} 1.
\]

This means that \( \mathcal{W}(nn) \) is the number of basis formulas in \( nn \) which have maximal order. Clearly, \( nn \neq 0 \Rightarrow \mathcal{W}(nn) \geq 1 \) and \( \theta(nn) = 0 \Rightarrow \mathcal{W}(nn) = 0 \) \( \vee \mathcal{W}(nn) = 1 \) (since \( B_0 \) has only one element).
Lemma 5.1:

Let $ss, tt \in ST$ with $tt \neq 0$. If $ss \neq \lambda tt$ for all $\lambda \in Q$, then

$$tt D(ss) - ss D(tt) \neq 0.$$ 

Proof:

Suppose $tt D(ss) - ss D(tt) = 0$, which implies $tt D(ss) - ss D(tt) \equiv 0$. Elementary differential calculus now yields the existence of a constant $\lambda$ such that $ss \equiv \lambda tt$. By Theorem 3.1, this contradicts with the assumption of the lemma.

Lemma 5.2:

Let $ss, tt, uu \in ST$ with $tt \neq 0$, $uu \neq 0$ and $NIT(uu)$. Then

$$tt^2 uu + tt D(ss) - ss D(tt) \neq 0.$$ 

Proof:

Suppose the converse holds, then also $tt^2 uu + tt D(ss) - ss D(tt) \equiv 0$. By elementary differential calculus we obtain $uu = -\frac{d}{dx} \frac{ss}{tt}$. If $ss$ can be reduced to a sumformula, this yields a contradiction since $uu \equiv 0$ and $NIT(uu)$ holds. Next, consider the case that $\frac{ss}{tt}$ cannot be reduced to a sumformula, i.e. $tt$ has factors which do not appear in $ss$. Using the unique factorization of $ss$ and $tt$ in prime factors, it is easily shown that in this case $\frac{d}{dx} \frac{ss}{tt}$ also cannot be written as a sumformula.

Recall that the lexicographical order on pairs of integers is defined by

$$(i, j) \preceq (k, \ell) \iff i < k \lor (i = k \land j < \ell).$$

Moreover, the set $\{(k, \ell) \mid (k, \ell) \succeq (0, 0)\}$ is a well-founded set on which the principle of induction holds, see for instance Barwise (1977).

Lemma 5.3:

For every $nn \in SN$ with $(0(nn), \omega(nn)) > (0, 1)$ there exists a $nn_1 \in SN$ with

$$(0, 1) \preceq (0(nn_1), \omega(nn_1)) < (0(nn), \omega(nn)) \quad (5.2)$$

and

$$(nn_1 \neq 0 \Rightarrow nn \neq 0). \quad (5.3)$$
Proof:
Let \( nn \in SN \) with \((O(nn),W(nn)) < (0,1)\). Since \( O(nn) = 0 \) and \( W(nn) > 1 \) is impossible, this means \((O(nn),W(nn)) \geq (1,1)\). Set \( k = O(nn) \) and \( \xi = W(nn) \).

Then we can write

\[
\Pi_k(nn) = \sum_{i=1}^{l} t_i b_i, \quad b_i \in B_k \text{ for } i = 1, \ldots, l.
\]

Define

\[
nn_1 = t_1 D(nn) - D(t_1)nn.
\]

Then \( nn_1 \in SN \) and (5.3) holds trivially. A simple calculation yields.

\[
\Pi_k(nn_1) = \sum_{i=1}^{l} (t_1 D(t_1) - t_i D(t_1))b_i.
\]

(5.4)

Clearly, this expression does not contain the basis formula \( b_j \). To prove (5.2) we consider two cases:

i) Suppose that for some \( j \) with \( 1 \leq j \leq l \) the sumterm \( t_j \) is not a multiple of \( t_1 \). Then Lemma 5.1 yields immediately that \( \Pi_k(nn_1) \) contains the basis formula \( b_j \). Hence \( O(nn_1) = k \) and \( 1 \leq W(nn_1) < l \).

ii) Suppose there exist constants \( \lambda_j \) such that \( t_j = \lambda_j t_1 \) for \( j = 1, \ldots, l \).

From (5.4) we now conclude that \( \Pi_k(nn_1) = 0 \), so \( O(nn_1) < k \). We shall now show that \( O(nn_1) = k - 1 \) and \( W(nn_1) \geq 1 \). Note that, since \( t_j \neq 0 \), also \( \lambda_j \neq 0 \) for \( j = 1, \ldots, l \). Let \( \Pi_{k-1}(nn) \) be given by

\[
\Pi_{k-1}(nn) = \sum_{i=1}^{m} s_i c_i, \quad c_i \in B_{k-1} \text{ for } i = 1, \ldots, m.
\]

Of course, \( nn \) does not necessarily contain formulas with order \( k-1 \), in that case \( m = 0 \). A straightforward calculation yields

\[
\Pi_{k-1}(nn_1) = t_1^2 \sum_{i=1}^{l} \lambda_i D(b_i) + \sum_{i=1}^{m} (t_1 D(s_i) - s_i D(t_1))c_i =
\]

\[
= t_1^2 \sum_{i=1}^{l} \lambda_i e_i + \sum_{i=1}^{m} (t_1 D(s_i) - s_i D(t_1))c_i,
\]

where \( e_i = D(b_i) \).
where we used that \( b_i \in B_k \) can be written as \( b_i = I(t_i, e_i) \) with \( t_i \in T \), \( e_i \in B_{k-1} \) and \( NIT(t_i) \) (\( i = 1, \ldots, \ell \)). We prove that this expression always contains the basis formula \( e_i \). Define

\[
uu = \sum_{i=1}^{\ell} \lambda_i t_i
\]

and

\[
ss = \sum_{i=1}^{m} ss_i
\]

Since all basis formulas \( b_i \) (\( i = 1, \ldots, \ell \)) are mutually different, the same must hold for the terms \( t_i \) which actually appear in the summation (5.5). Hence, \( uu \neq 0 \) and \( NIT(uu) \) holds. Note that, because all basis formulas \( c_i \) are different, the summation (5.6) takes place over at most one value of \( i \). The 'coefficient' of \( e_1 \) in \( \Pi_{k-1}(nn_1) \) can now be written as

\[
\begin{pmatrix} t \end{pmatrix}^2 uu + \begin{pmatrix} t \end{pmatrix} D(ss) - ss D\begin{pmatrix} t \end{pmatrix}
\]

Lemma 5.2 yields that this sum term does not cancel out, so \( \Pi_{k-1}(nn_1) \) always contains the basis formula \( e_1 \). Hence \( \theta(nn_1) = k - 1 \) and \( \omega(nn_1) \geq 1 \). 

Now the uniqueness of the normal forms is easily shown.

**Theorem 5.1:**

For all sumformulas \( nn \in SN \):

\( nn = 0 \iff nn \equiv 0 \).

**Proof:**

Of course, we only have to show \( nn \equiv 0 \iff nn = 0 \), or equivalently \( nn \neq 0 \iff nn \equiv 0 \). From the definitions of order and width we see that this corresponds to proving that \( nn \equiv 0 \) for all \( nn \in SN \) with \( (\theta(nn),\omega(nn)) \geq (0,1) \).

This is easily shown using induction with respect to the pair \( (\theta(nn),\omega(nn)) \) under the lexicographical order. The induction basis \( (\theta(nn),\omega(nn)) = (0,1) \)
follows from Theorem 3.1, while the induction step is obtained from Lemma 5.3.

Several equivalent formulations of this theorem can be given. For instance

\[ n_{n_1} = n_{n_2} \Leftrightarrow n_{n_1} \equiv n_{n_2} \]

for all \( n_{n_1}, n_{n_2} \in S_N \). Also, if for \( i = 1, \ldots, m \) the \( n_i \in N \) are mutually different and \( \lambda_i \in Q \) (possibly \( \lambda_i = 0 \)), then

\[ \sum_{i=1}^{m} \lambda_i n_i \equiv 0 \Leftrightarrow (\forall i: 1 \leq i \leq m: \lambda_i = 0) . \]

6. THE PREDICATE NIT AND THE MAPPINGS \( \text{Int} \) AND Rest

The results given in Sections 4 and 5, i.e. the normalizing mapping \( M \) and the uniqueness of the normal forms, have been derived under the assumption that the hypothesis \( H \) (Section 3) holds. In this section we shall show that this is indeed the case, i.e. we shall construct a predicate \( \text{NIT} \) on \( S_T \) and mappings \( \text{Int}, \text{Rest}: S_T \to S_T \) such that (3.2)-(3.5) hold. The construction of \( \text{NIT}, \text{Int} \) and \( \text{Rest} \) may look technical, but it is in fact only a matter of partial integration. First we describe the predicate \( \text{NIT} \). Consider a term \( t \), i.e. a product of functions \( a_1, \ldots, a_N \) and their derivatives. Let \( h_i(t) \) be the highest derivative of \( a_i \) which occurs in \( t \) and let \( p_i(t) \) be the power of this derivative. If \( a_i \) and its derivatives do not occur in \( t \), then \( h_i(t) = -1 \) and \( p_i(t) = 0 \). Further we define

\[ H(t) = (\operatorname{MAX} i: 1 \leq i \leq N: h_i(t)) , \]

\[ P(t) = \sum_{i=1}^{N} \frac{p_i(t)}{h_i(t)=H(t)} \]

and

\[ J(t) = (\operatorname{MIN} i: 1 \leq i \leq N \land h_i(t) = H(t): i) . \]
So \( H(t) \) is the highest derivative, \( P(t) \) is the number of factors which have this derivative and \( J(t) \) is the lowest function number which has derivative \( H(t) \) in the term \( t \). For instance, if \( t = a_1^2 x a_2^3 x^3 a_3 x^2 \), then \( H(t) = 2 \), \( P(t) = 4 \) and \( J(t) = 2 \). The predicate \( \text{NIT}(t) \) is now defined by

\[
\text{NIT}(t) \equiv t = 1 \lor H(t) = 0 \lor P(t) \geq 2 \lor 
\left( \exists i: 1 \leq i < J(t): h_i(t) = H(t) - 1 \right). 
\tag{6.1}
\]

So \( \text{NIT}(t) \) holds if i) \( t = 1 \) or ii) \( t \) does not contain derivatives or iii) the number of factors in \( t \) which have the highest derivative is at least 2 or iv) there exists a factor in \( t \) with derivative \( H(t) - 1 \) and a function number less than \( J(t) \). For instance, the predicates \( \text{NIT}(1) \), \( \text{NIT}(a_1 a_2^3 a_4) \), \( \text{NIT}(a_1 a_2^2 _{xxx} a_3 _{xx}) \), \( \text{NIT}(a_1 a_2^3 _{xx} a_3 _{xx}) \) and \( \text{NIT}(a_1 a_2^1 x) \) hold, but \( \text{NIT}(a_1 a_2) \) does not hold.

For sumterms we define

\[
\text{NIT}\left( \sum_{i=1}^{m} \lambda_i t_i \right) \equiv (\forall i: 1 \leq i \leq m: \text{NIT}(t_i)),
\]

so (3.2) trivially holds. Next we prove (3.3). Let \( uu = \sum_{i=1}^{m} \lambda_i t_i \) be a non-vanishing sumterm such that \( \text{NIT}(uu) \) holds and suppose there exists a sum-term \( ss \) such that \( I(uu) \equiv ss \), or equivalently

\[
uu = D(ss).
\tag{6.2}
\]

Let \( d \) be the highest derivative which occurs in \( ss \) and let \( \ell \) be the lowest function number in \( ss \) for which this derivative occurs. Then by considering all terms in \( ss \) in which the \( d \)-th derivative of \( a_\ell \) occurs, it is easily seen that (6.2) leads to a contradiction with \( \text{NIT}(uu) \). Hence (3.3) holds.

The mappings \( \text{Int} \) and \( \text{Rest} \) are defined by giving an algorithm that, for a sumterm \( tt \), computes \( re = \text{Rest}(tt) \) and \( in = \text{Int}(tt) \). Informally the algorithm works as follows. Terms in \( tt \) for which \( \text{NIT} \) holds are transferred to \( re \). For terms in \( tt \) for which \( \text{NIT} \) does not hold, a partial integration can be performed. More precisely, if \( \text{NIT}(t) \) does not hold, then (6.1) implies that \( t \) can be written as

\[
t = a_j^{m (h-1) x} a_j^{h^x} s, \tag{6.3}
\]
where \( h = H(t) \), \( j = J(t) \), \( m \geq 0 \) and \( s \) is a term with \( H(s) < h \) and if \( H(s) = h - 1 \), then \( J(s) > j \) \((a_{j_{kx}} = \partial^k(a_j))\). Partial integration (2.2) now yields

\[
I(t) = \frac{1}{m+1} a_{j}^{m+1} (h-1)x s - I\left( \frac{1}{m+1} a_{j}^{m+1} (h-1)x D(s) \right).
\]

The first term in the right-hand side is now added to \( in \), while the terms in \( tt_1 = - \frac{1}{m+1} a_{j}^{m+1} (h-1)x D(s) \) are again added to \( tt \) (all with appropriate coefficients). From the properties of \( s \) mentioned above, it is easily seen that each term in \( tt_1 \) has i) a highest derivative less than \( h \) or ii) a highest derivative equal to \( h \), but then this derivative can only appear for functions \( a_i \) with \( i > j \). Hence, by removing \( t \) from \( tt \) and (in case of \( \neg NIT(t) \)) adding the terms in \( tt_1 \) to \( tt \), the highest derivatives in \( tt \) decrease or stay equal and shift to functions with higher numbers. So it is possible to repeat the steps above until \( tt = 0 \). We now give the formal description of the algorithm. Its correctness follows from the loop invariant

\[
P: I(TT) = I(tt) + in + I(re) \wedge NIT(re).
\]

tt := TT; re := 0; in := 0;
{invariant P}
while \( tt \neq 0 \) do
  let \( t \) be a term in \( tt \) with coefficient \( \lambda \);
  \( tt := tt - \lambda t \);
  if \( NIT(t) \) then \( re := re + \lambda t \) \{P\}
  else
    compute \( s \) such that (6.3) holds;
    \( in := in + \frac{\lambda}{m+1} a_{j}^{m+1} (h-1)x s \);
    \( tt := tt - \frac{\lambda}{m+1} a_{j}^{m+1} (h-1)x D(s) \) \{P\}
  fi
{P}
od \{P \wedge tt = 0, so \( I(TT) = in + I(re) \wedge NIT(re) \)\}
It is possible to replace the informal arguments for the termination of the repetition given above by a more formal termination proof using a variant function, but we shall not work out that here. Clearly the mappings $\text{Int}$ and $\text{Rest}$, defined by $\text{Int}(TT) = \text{in}$ and $\text{Rest}(TT) = \text{re}$ satisfy (3.4) and (3.5). Thus we have shown that the hypothesis $H$ can be satisfied.

Note that in this section we used in fact an order on the functions $a_1, \ldots, a_N$. Of course, any other order could also be used. Hence for sumformulas which consist of $N$ functions there exist in fact $N!$ different normal forms.

7. CONCLUDING REMARKS

The normalizing algorithm described in Sections 4 and 6 can easily be implemented in a suitable formula manipulation system. An implementation in the MUSIMP system is straightforward and can be used to perform the calculations mentioned in the introduction. One of us (J.C.F.W.) constructed a PASCAL implementation for the case $N = 1$. However, the resulting program turned out to be too slow for practical computations.

In the process of computing a normal form only the relations (2.2), (2.3) and (3.4) are used. Moreover, the left-hand side of these relations is always replaced by the right-hand side. Hence we can consider the set of sumformulas as a term rewriting system with reduction rules (2.2), (2.3) and (3.4). In this approach the mapping $M$ describes a reduction strategy which always leads to a sumformula in normal form. Note the similarity with the probably most well-known term rewriting system, the Lambda calculus. Possibly there exist reduction strategies which lead to the normal form in less steps than the strategy used here. This question is investigated at the moment.

Acknowledgement: We thank I.J.M. Canjels for the formulation of the predicate $\text{NIT}$ and for implementing the mappings $\text{Int}$ and $\text{Rest}$ in the MUSIMP system. Moreover, we thank J.M. Kloosterman for writing a MUSIMP implementation of the normalizing mapping $M$. 
REFERENCES


In this series appeared:

<table>
<thead>
<tr>
<th>No.</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>85/01</td>
<td>R.H. Mak</td>
<td>The formal specification and derivation of CMOS-circuits</td>
</tr>
<tr>
<td>85/02</td>
<td>W.M.C.J. van Overveld</td>
<td>On arithmetic operations with M-out-of-N-codes</td>
</tr>
<tr>
<td>85/03</td>
<td>W.J.M. Lemmens</td>
<td>Use of a computer for evaluation of flow films</td>
</tr>
<tr>
<td>85/04</td>
<td>T. Verhoeff, H.M.J.L. Schols</td>
<td>Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate</td>
</tr>
<tr>
<td>86/01</td>
<td>R. Koymans</td>
<td>Specifying message passing and real-time systems</td>
</tr>
<tr>
<td>86/02</td>
<td>G.A. Bussing, K.M. van Hee, M. Voorhoeve</td>
<td>ELISA, A language for formal specifications of information systems</td>
</tr>
<tr>
<td>86/03</td>
<td>Rob Hoogerwoord</td>
<td>Some reflections on the implementation of trace structures</td>
</tr>
<tr>
<td>86/04</td>
<td>G.J. Houben, J. Paredaens, K.M. van Hee</td>
<td>The partition of an information system in several parallel systems</td>
</tr>
<tr>
<td>86/05</td>
<td>Jan L.G. Dietz, Kees M. van Hee</td>
<td>A framework for the conceptual modeling of discrete dynamic systems</td>
</tr>
<tr>
<td>86/06</td>
<td>Tom Verhoeoff</td>
<td>Nondeterminism and divergence created by concealment in CSP</td>
</tr>
<tr>
<td>86/07</td>
<td>R. Gerth, L. Shira</td>
<td>On proving communication closedness of distributed layers</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>86/09</td>
<td>C. Huizing, R. Gerth, W.P. de Roever</td>
<td>Full abstraction of a real-time denotational semantics for an OCCAM-like language</td>
</tr>
<tr>
<td>86/10</td>
<td>J. Hooman</td>
<td>A compositional proof theory for real-time distributed message passing</td>
</tr>
<tr>
<td>86/11</td>
<td>W.P. de Roever</td>
<td>Questions to Robin Milner - A responders commentary (IFIP86)</td>
</tr>
<tr>
<td>86/12</td>
<td>A. Boucher, R. Gerth</td>
<td>A timed failures model for extended communicating processes</td>
</tr>
<tr>
<td>86/14</td>
<td>R. Koymans</td>
<td>Specifying passing systems requires extending temporal logic</td>
</tr>
<tr>
<td>87/01</td>
<td>R. Gerth</td>
<td>On the existence of a sound and complete axiomatizations of the monitor concept</td>
</tr>
<tr>
<td>87/02</td>
<td>Simon J. Klaver, Chris F.M. Verberne</td>
<td>Federatieve Databases</td>
</tr>
<tr>
<td>87/03</td>
<td>G.J. Houben, J. Paredaens</td>
<td>A formal approach to distributed information systems</td>
</tr>
<tr>
<td>87/04</td>
<td>T. Verhoeff</td>
<td>Delay-insensitive codes - An overview</td>
</tr>
<tr>
<td>87/05</td>
<td>R. Kuiper</td>
<td>Enforcing non-determinism via linear time temporal logic specification</td>
</tr>
</tbody>
</table>
87/06 R. Koymans
Temporele logica specificatie van message passing
en real-time systemen (in Dutch)

87/07 R. Koymans
Specifying message passing and real-time
systems with real-time temporal logic

87/08 H.M.J.L. Schols
The maximum number of states after projection

87/09 J. Kalisvaart
L.R.A. Kessener
W.J.M. Lemmens
M.L.P van Lierop
F.J. Peters
H.M.M. van de Wetering
Language extensions to study structures
for raster graphics

87/10 T. Verhoeff
Three families of maximally nondeterministic
automata

87/11 P. Lemmens
Eldorado ins and outs.
Specifications of a data base management
toolkit according to the functional model

87/12 K.M. van Hee
A. Lapinski
OR and AI approaches to decision support
systems

87/13 J. van der Woude
Playing with patterns, searching for strings

87/14 J. Hooman
A compositional proof system for an occam-
like real-time language

87/15 G. Huizing
R. Gerth
W.P. de Roever
A compositional semantics for statecharts

87/16 H.M.M. ten Eikelder
J.C.F. Wilmont
Normal forms for a class of formulas

87/17 K.M. van Hee
G.J. Houben
J.L.G. Dietz
Modelling of discrete dynamic systems
framework and examples
<table>
<thead>
<tr>
<th>87/18</th>
<th>C.W.A.M. van Overveld</th>
<th>An integer algorithm for rendering curved surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>87/19</td>
<td>A.J. Seebregts</td>
<td>Optimalisering van file allocatie in gedistribueerde database systemen</td>
</tr>
</tbody>
</table>