Normal forms for a class of formulas

Normal forms for a class of formulas

by

H.M.M. ten Eikelder

J.C.F. Wilmont

87/16

december 1987

COMPUTING SCIENCE NOTES

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.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands

editor: F.A.J. van Neerven

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 a1,...,aN, 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 a1,...,aN: $\mathbb{R} \to \mathbb{R}$ and their derivatives, and integration operators I. Different expressions of this type can have the same meaning. For instance, if differentiation is denoted by a subscript x, the expressions $I(a1\, a2_x) + I(a1_x\, a2)$ and a1 a2 have the same meaning (under appropriate boundary conditions and definition of 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 a1,...,aN 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':

$$1 \text{ (empty product)}\ ,\quad a1_{xx}\, a2\, a3_x^2\ ,\quad a1\, a4_{xx}^3\ .$$

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

```
f ::= t        (t ∈ T)
f ::= I(f)     (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,

$$a1\, a4_{xx}^3\ ,\quad a1\, a2\, I(a1_x\, I(a2_x^2) I(I(a2_x\, a3)))\ .$$

The set of sumformulas SF is defined by

$$SF = \left\{ \sum_{i=1}^{m} \lambda_i f_i \;\middle|\; m \geq 0,\ (\forall i\colon 1 \leq i \leq m\colon \lambda_i \in Q,\ f_i \in F) \right\} ,$$

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

$$I(a1_x\, I(a2\, a2_x) I(a2_x\, a3)) - \frac{1}{3}\, a1\, a2^3\, a3 + I(a1\, a2\, a2_x\, I(a2_x\, a3)) +$$
$$+ \frac{1}{2}\, a1\, a2^2\, I(a2\, a3_x) - \frac{1}{6}\, I(a1\, a2^3\, a3_x) - \frac{1}{6}\, I(a1_x\, a2^3\, a3)\ . \qquad (2.1)$$

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 $a1, \dots, aN \in C$ and $I(h)(x) = {}_{-\infty}\!\int^{x} h(y)dy$, 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 $C^N \to \hat{C}$, 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, $\frac{1}{3}\, a1\, a1_x\, I(a2)$ has the same meaning as $\frac{2}{3}\, a1\, I(a2) a1_x - \frac{1}{3}\, I(a2) a1\, a1_x$. 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 $g1 = g2$ means that g1 and g2 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 λ_i do not vanish and that $f_i \neq f_j$ for $i \neq j$.

If two (sum)formulas g1 and g2 have the same meaning we shall write $g1 \overset{m}{=} g2$. Clearly, $\overset{m}{=}$ is an equivalence relation. For instance, $I(a1_x) \overset{m}{=} a1$, but $I(a1_x) \neq a1$. More complicated different sumformulas with the same meaning can easily be found using partial integration.

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

$$I(f\mathcal{D}(g)) \overset{m}{=} fg - I(g\mathcal{D}(f)) , \qquad (2.2)$$

or equivalently

$$I(f)I(g) \overset{m}{=} I(fI(g)) + I(gI(f)) \ . \tag{2.3}$$

An elementary computation using these relations shows that (2.1) has the same meaning as 0. 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 SN of the set of sumformulas SF such that

i) every sumformula in SF can be transformed to a sumformula in SN with the same meaning,
ii) two sumformulas in SN have the same meaning if and only if they are equal.

Algebraically SN is isomorphic with $SF / \overset{m}{=}$, 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 ST is defined by

$$ST = \left\{ \sum_{i=1}^{m} \lambda_i t_i \mid m \geq 0,\ (\forall i\colon 1 \leq i \leq m\colon \lambda_i \in Q,\ t_i \in T) \right\} .$$

Then, $T \subset ST \subset SF$ (also $T \subset F \subset SF$). 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:

$$t, t_1, t_2, \ldots,\ s, s_1, s_2, \ldots \in T \ ,$$

$$tt, tt_1, tt_2, \ldots,\ ss, ss_1, ss_2, \ldots,\ uu \in ST \ ,$$

$$f, f_1, f_2, \ldots \in F \ ,$$

$$ff \in SF \ ,$$

$$\lambda, \lambda_1, \lambda_2, \ldots \in Q \ .$$

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 $\Leftrightarrow$ tt $\overset{m}{=}$ 0.

Proof:

Any sumterm can be considered as a polynomial in a number of variables which is a finite subset of $\{a1, a1_x, a1_{xx}, \ldots, aN, aN_x, \ldots\}$. Moreover, every set of values for these variables can be obtained as the corresponding derivatives of functions $a1, a2, \ldots, aN \in C$ 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

$$I(a1\, a2_{xxxx} + a1_{xx}\, a2_{xx}) \overset{m}{=} a1\, a2_{xxx} - a1_x\, a2_{xx} + 2I(a1_{xx}\, a2_{xx}) .$$

This suggests to try $tt_1 + I(tt_2)$ as normal form for I(tt), where the sumterms 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

$$tt_1 - ss_1 \overset{m}{=} I(ss_2 - tt_2) \tag{3.1}$$

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

$$NIT\left(\sum_{i=1}^{m} \lambda_i t_i\right) = (\forall i: 1 \leq i \leq m: NIT(t_i)) \tag{3.2}$$

and for uu $\neq 0$

$$NIT(uu) \Rightarrow (\forall ss: ss \in ST: I(uu) \overset{m}{\neq} ss) \ . \tag{3.3}$$

So, if NIT(uu) holds and uu $\neq 0$, then uu cannot be the derivative of a sumterm. Clearly, if in (3.1) $NIT(ss_2)$ and $NIT(tt_2)$ hold, then also $NIT(ss_2 - tt_2)$ 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 $NIT(tt_2)$ 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 NIT on ST that satisfies (3.2) and (3.3) and there exist mappings Int: ST $\rightarrow$ ST and $Rest$: ST $\rightarrow$ ST such that

$$I(tt) \overset{m}{=} Int(tt) + I(Rest(tt)) \tag{3.4}$$

and

$$NIT(Rest(tt)) \ . \tag{3.5}$$

□

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 NIT and mappings Int and $Rest$ which satisfy the hypothesis H. Note that D, Int, $Rest$ (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 SN of the set of sumformulas SF. The main result of this section is Theorem 4.1, which states that for every sumformula in SF a sumformula in 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

$$B_0 = \{1\} ,$$

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

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, $T \subsetneqq N \subsetneqq F$. The order $\mathcal{O}$ of a formula in N is defined by:

$$\mathcal{O}(tb) = k \quad \text{if } b \in B_k .$$

So, $\mathcal{O}(n)$ 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 $\mathit{NIT}(t_i)$ for $i = 1,\ldots,k$.

The set SN of sumformulas in normal form is defined by

$$SN = \left\{ \sum_{i=1}^{m} \lambda_i n_i \mid m \geq 0,\ (\forall i\colon 1 \leq i \leq m\colon \lambda_i \in Q,\ n_i \in N) \right\} .$$

Then $ST \subsetneqq SN \subsetneqq SF$. We generalize the notion of order to SN by

$$\mathcal{O}\left(\sum_{i=1}^{m} \lambda_i n_i\right) = \begin{cases} 0 & \text{if } m = 0 , \\ (\underline{\mathrm{MAX}}\ i\colon 1 \leq i \leq m\colon \mathcal{O}(n_i)) & \text{if } m \geq 1 . \end{cases}$$

By gathering formulas which have the same basis formula, every sumformula nn $\in$ SN can be written as

$$nn = \sum_{i=1}^{m} tt_i b_i \ , \qquad tt_i \in ST,\ b_i \in B \text{ for } i = 1,\dots,m \ , \tag{4.1}$$

where the basis formulas b_i are mutually different and $tt_i \neq 0$ for $i = 1,\dots,m$. 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

$$b, b_1, b_2, \dots,\ c, c_1, c_2, \dots,\ e_1, e_2, \dots \in B \ ,$$

$$n, n_1, n_2, \dots \in N \ ,$$

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

In the remaining part of this section we construct a mapping M: SF $\to$ 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

$$\begin{aligned} I(tt\,b) &\overset{m}{=} I((\mathcal{D}(\mathit{Int}(tt) + \mathit{Rest}(tt))b) \\ &\overset{m}{=} \mathit{Int}(tt)b - I(\mathit{Int}(tt)\mathcal{D}(b)) + I(\mathit{Rest}(tt)b) \ . \end{aligned} \tag{4.2}$$

The first and, since $NIT(\mathit{Rest}(tt))$ holds, the last expression in (4.2) consists of formulas in normal form. If b $\in B_0$, then $\mathcal{D}(b) = \mathcal{D}(1) = 0$ 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 $I(\mathit{Int}(tt)\mathcal{D}(b))$ is known. Since $O(\mathit{Int}(tt)\mathcal{D}(b)) = k-1$ and $O(ttb) = k$, we can use recursion to compute the normal form of I(ttb). Define M_1: SN $\to$ SN by

$$\begin{aligned} M_1(\mathrm{tt}) &= \mathit{Int}(\mathrm{tt}) + \mathrm{I}(\mathit{Rest}(\mathrm{tt})) \ , \\ M_1(\mathrm{tt\,b}) &= \mathit{Int}(\mathrm{tt})\mathrm{b} + \mathrm{I}(\mathit{Rest}(\mathrm{tt}))\mathrm{b} \qquad (\mathrm{b} \in \mathrm{B} \setminus \mathrm{B}_0) \\ &\qquad - M_1(\mathit{Int}(\mathrm{tt})\mathcal{D}(\mathrm{b})) \ , \\ M_1(\sum_i \mathrm{tt}_i \mathrm{b}_i) &= \sum_i M_1(\mathrm{tt}_i \mathrm{b}_i) \ . \end{aligned} \tag{4.3}$$

The proof of the following lemma is now almost trivial.

Lemma 4.1:
For all nn $\in$ SN:

$$M_1(\mathrm{nn}) \overset{\mathrm{m}}{=} \mathrm{I}(\mathrm{nn}) \ . \qquad \square$$

So, if nn is in normal form, M_1(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

$$\mathrm{I(tb)I(sc)} \overset{\mathrm{m}}{=} \mathrm{I(tbI(sc))} + \mathrm{I(scI(tb))} \ . \tag{4.4}$$

Suppose that normal forms for the products of basis formulas bI(sc), respectively cI(tb) are known. Then, using the mapping M_1, a normal form for the product I(tb)I(sc) can easily be computed from (4.4). Since

$$\mathcal{O}(\mathrm{b}) + \mathcal{O}(\mathrm{I(sc)}) = \mathcal{O}(\mathrm{c}) + \mathcal{O}(\mathrm{I(tb)}) < \mathcal{O}(\mathrm{I(tb)}) + \mathcal{O}(\mathrm{I(sc)}) \ ,$$

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

$$\begin{aligned} M_2(1,\mathrm{c}) &= \mathrm{c} \ , \\ M_2(\mathrm{b},1) &= \mathrm{b} \ , \\ M_2(\mathrm{I(tb)},\mathrm{I(sc)}) &= M_1(\mathrm{t}\,M_2(\mathrm{b},\mathrm{I(sc)})) + M_1(\mathrm{s}\,M_2(\mathrm{c},\mathrm{I(tb)})) \ , \\ M_2(\sum_i \mathrm{tt}_i \mathrm{b}_i, \sum_j \mathrm{ss}_j \mathrm{c}_j) &= \sum_{i,j} \mathrm{tt}_i \mathrm{ss}_j \, M_2(\mathrm{b}_i,\mathrm{c}_j) \ . \end{aligned} \tag{4.5}$$

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:

$$M_2(\text{nn1},\text{nn2}) \overset{m}{=} \text{nn1 nn2} .$$ □

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

$$\begin{aligned} M(t) &= t , \\ M(I(f)) &= M_1(M(f)) , \\ M(f_1 f_2) &= M_2(M(f_1),M(f_2)) , \\ M\left(\sum_{i=1}^{m} \lambda_i f_i\right) &= \sum_{i=1}^{m} \lambda_i M(f_i) . \end{aligned} \tag{4.6}$$

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:

$$M(\text{ff}) \overset{m}{=} \text{ff} .$$

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

$$\mathrm{nn} = \sum_{i=1}^{m} \mathrm{tt}_i\, b_i\ , \tag{5.1}$$

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

$$\Pi_k(\mathrm{nn}) = \sum_{\substack{i=1\\ \mathcal{O}(b_i)=k}}^{m} \mathrm{tt}_i\, b_i\ .$$

So, Π_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

$$\mathcal{W}(\mathrm{nn}) = \sum_{\substack{i=1\\ \mathcal{O}(b_i)=\mathcal{O}(\mathrm{nn})}}^{m} 1\ .$$

This means that $\mathcal{W}$(nn) is the number of basis formulas in nn which have maximal order. Clearly, nn $\neq 0 \Leftrightarrow \mathcal{W}(\mathrm{nn}) \geq 1$ and $\mathcal{O}(\mathrm{nn}) = 0 \Rightarrow \mathcal{W}(\mathrm{nn}) = 0 \vee \mathcal{W}(\mathrm{nn}) = 1$ (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*

$$\mathrm{tt}\,\mathcal{D}(\mathrm{ss}) - \mathrm{ss}\,\mathcal{D}(\mathrm{tt}) \neq 0 \ .$$

Proof:

Suppose tt$\mathcal{D}$(ss) - ss$\mathcal{D}$(tt) = 0, which implies tt$\mathcal{D}$(ss) - ss$\mathcal{D}$(tt) $\overset{m}{=} 0$. Elementary differential calculus now yields the existence of a constant λ such that ss $\overset{m}{=} \lambda$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*

$$\mathrm{tt}^2\,\mathrm{uu} + \mathrm{tt}\,\mathcal{D}(\mathrm{ss}) - \mathrm{ss}\,\mathcal{D}(\mathrm{tt}) \neq 0 \ .$$

Proof:

Suppose the converse holds, then also tt^{2}uu + tt$\mathcal{D}$(ss) - ss$\mathcal{D}$(tt) $\overset{m}{=} 0$. By elementary differential calculus we obtain uu $\overset{m}{=} -\frac{d}{dx}(\frac{ss}{tt})$. If $\frac{ss}{tt}$ can be reduced to a sumformula, this yields a contradiction since uu $\neq 0$ 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

$$(i,j) \leq (k,\ell) \Leftrightarrow i < k \vee (i = k \wedge j < \ell) \ .$$

Moreover, the set $\{(k,\ell) \mid (k,\ell) \geq (0,0)\}$ 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* $(\mathcal{O}(\mathrm{nn}),\mathcal{W}(\mathrm{nn})) > (0,1)$ *there exists a* nn$_1$ $\in$ SN *with*

$$(0,1) \leq (\mathcal{O}(\mathrm{nn}_1),\mathcal{W}(\mathrm{nn}_1)) < (\mathcal{O}(\mathrm{nn}),\mathcal{W}(\mathrm{nn})) \tag{5.2}$$

and

$$\mathrm{nn}_1 \overset{m}{\neq} 0 \Rightarrow \mathrm{nn} \overset{m}{\neq} 0 \ . \tag{5.3}$$

Proof:

Let nn $\in$ SN with $(\mathcal{O}(nn),\mathcal{W}(nn)) > (0,1)$. Since $\mathcal{O}(nn) = 0$ and $\mathcal{W}(nn) > 1$ is impossible, this means $(\mathcal{O}(nn),\mathcal{W}(nn)) \geq (1,1)$. Set $k = \mathcal{O}(nn)$ and $\ell = \mathcal{W}(nn)$. Then we can write

$$\Pi_k(nn) = \sum_{i=1}^{\ell} tt_i b_i \ , \qquad b_i \in B_k \text{ for } i = 1,\ldots,\ell \ .$$

Define

$$nn_1 = tt_1 \mathcal{D}(nn) - \mathcal{D}(tt_1) nn \ .$$

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

$$\Pi_k(nn_1) = \sum_{i=1}^{\ell} (tt_1 \mathcal{D}(tt_i) - tt_i \mathcal{D}(tt_1)) b_i \ . \tag{5.4}$$

Clearly, this expression does not contain the basis formula b_1. To prove (5.2) we consider two cases:

i) Suppose that for some j with $1 \leq j \leq \ell$ the sumterm tt_j is not a multiple of tt_1. Then Lemma 5.1 yields immediately that $\Pi_k(nn_1)$ contains the basis formula b_j. Hence $\mathcal{O}(nn_1) = k$ and $1 \leq \mathcal{W}(nn_1) < \ell$.

ii) Suppose there exist constants λ_j such that $tt_j = \lambda_j tt_1$ for $j = 1,\ldots,\ell$. From (5.4) we now conclude that $\Pi_k(nn_1) = 0$, so $\mathcal{O}(nn_1) < k$. We shall now show that $\mathcal{O}(nn_1) = k-1$ and $\mathcal{W}(nn_1) \geq 1$. Note that, since $tt_j \neq 0$, also $\lambda_j \neq 0$ for $j = 1,\ldots,\ell$. Let $\Pi_{k-1}(nn)$ be given by

$$\Pi_{k-1}(nn) = \sum_{i=1}^{m} ss_i c_i \ , \qquad c_i \in B_{k-1} \text{ for } i = 1,\ldots,m \ .$$

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

$$\Pi_{k-1}(nn_1) = tt_1^2 \sum_{i=1}^{\ell} \lambda_i \mathcal{D}(b_i) + \sum_{i=1}^{m} (tt_1 \mathcal{D}(ss_i) - ss_i \mathcal{D}(tt_1)) c_i =$$

$$= tt_1^2 \sum_{i=1}^{\ell} \lambda_i t_i e_i + \sum_{i=1}^{m} (tt_1 \mathcal{D}(ss_i) - ss_i \mathcal{D}(tt_1)) c_i \ ,$$

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 $NIT(t_i)$ $(i = 1,\dots,\ell)$. We prove that this expression always contains the basis formula e_1. Define

$$uu = \sum_{\substack{i=1 \\ e_i=e_1}}^{\ell} \lambda_i t_i \tag{5.5}$$

and

$$ss = \sum_{\substack{i=1 \\ c_i=e_1}}^{m} ss_i \ . \tag{5.6}$$

Since all basis formulas b_i $(i = 1,\dots,\ell)$ 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

$$tt_1^2\, uu + tt_1 \mathcal{D}(ss) - ss\, \mathcal{D}(tt_1) \ .$$

Lemma 5.2 yields that this sumterm does not cancel out, so $\Pi_{k-1}(nn_1)$ always contains the basis formula e_1. Hence $\mathcal{O}(nn_1) = k-1$ and $\mathcal{W}(nn_1) \geq 1$. □

Now the uniqueness of the normal forms is easily shown.

<u>Theorem 5.1</u>:
For all sumformulas $nn \in SN$:

$$nn = 0 \Leftrightarrow nn \overset{m}{=} 0 \ .$$

<u>Proof</u>:
Of course, we only have to show $nn \overset{m}{=} 0 \Rightarrow nn = 0$, or equivalently $nn \neq 0 \Rightarrow nn \overset{m}{\neq} 0$. From the definitions of order and width we see that this corresponds to proving that $nn \overset{m}{\neq} 0$ for all $nn \in SN$ with $(\mathcal{O}(nn),\mathcal{W}(nn)) \geq (0,1)$. This is easily shown using induction with respect to the pair $(\mathcal{O}(nn),\mathcal{W}(nn))$ under the lexicographical order. The induction basis $(\mathcal{O}(nn),\mathcal{W}(nn)) = (0,1)$

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

$$nn_1 = nn_2 \Leftrightarrow nn_1 \overset{m}{=} nn_2 ,$$

for all $nn_1, nn_2 \in SN$. 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

$$\sum_{i=1}^{m} \lambda_i n_i \overset{m}{=} 0 \Rightarrow (\forall i\colon 1 \leq i \leq m\colon \lambda_i = 0) .$$

6. THE PREDICATE NIT AND THE MAPPINGS 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 *NIT* on ST and mappings *Int*, *Rest*: ST → ST such that (3.2)-(3.5) hold. The construction of *NIT*, *Int* and *Rest* may look technical, but it is in fact only a matter of partial integration. First we describe the predicate *NIT*. Consider a term t, i.e. a product of functions a1,...,aN and their derivatives. Let $h_i(t)$ be the highest derivative of ai which occurs in t and let $p_i(t)$ be the power of this derivative. If ai and its derivatives do not occur in t, then $h_i(t) = -1$ and $p_i(t) = 0$. Further we define

$$H(t) = (\underline{\text{MAX}}\ i\colon 1 \leq i \leq N\colon h_i(t)) ,$$

$$P(t) = \sum_{\substack{i=1 \\ h_i(t)=H(t)}}^{N} p_i(t)$$

and

$$J(t) = (\underline{\text{MIN}}\ i\colon 1 \leq i \leq N \wedge h_i(t) = H(t)\colon i) .$$

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 $t = a1_x^2\, a2_{xx}^3\, a3_x\, a3_{xx}$, then $H(t) = 2$, $P(t) = 4$ and $J(t) = 2$. The predicate $NIT(t)$ is now defined by

$$NIT(t) \equiv t = 1 \vee H(t) = 0 \vee P(t) \geq 2 \vee$$
$$(\exists i\colon 1 \leq i < J(t)\colon h_i(t) = H(t) - 1) \ . \tag{6.1}$$

So $NIT(t)$ 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 $NIT(1)$, $NIT(a1\, a2^3\, a4)$, $NIT(a1\, a2_{xxx}^2\, a3_{xx})$, $NIT(a1_{xx}\, a2_x^3\, a3_{xx})$ and $NIT(a1\, a2_x)$ hold, but $NIT(a1_x\, a2)$ does not hold.

For sumterms we define

$$NIT\left(\sum_{i=1}^{m} \lambda_i t_i\right) \equiv (\forall i\colon 1 \leq i \leq m\colon NIT(t_i)) \ ,$$

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 $NIT(uu)$ holds and suppose there exists a sumterm ss such that $I(uu) \overset{m}{=} ss$, or equivalently

$$uu = \mathcal{D}(ss) \ . \tag{6.2}$$

Let d be the highest derivative which occurs in ss and let ℓ 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 $NIT(uu)$. Hence (3.3) holds.

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

$$t = aj_{(h-1)x}^m\, aj_{hx}\, s \ , \tag{6.3}$$

where $h = \mathcal{H}(t)$, $j = \mathcal{J}(t)$, $m \geq 0$ and s is a term with $\mathcal{H}(s) < h$ and if $\mathcal{H}(s) = h-1$, then $\mathcal{J}(s) > j$ $(aj_{kx} = \mathcal{D}^k(aj))$. Partial integration (2.2) now yields

$$I(t) \overset{m}{=} \frac{1}{m+1}\, aj_{(h-1)x}^{m+1}\, s - I\left(\frac{1}{m+1}\, aj_{(h-1)x}^{m+1}\, \mathcal{D}(s)\right) .$$

The first term in the right-hand side is now added to in, while the terms in $tt1 = -\frac{1}{m+1}\, aj_{(h-1)x}^{m+1}\, \mathcal{D}(s)$ are again added to tt (all with appropriate coefficients). From the properties of s mentioned above, it is easily seen that each term in tt1 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 ai with $i > j$. Hence, by removing t from tt and (in case of $\neg NIT(t)$) adding the terms in tt1 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

$$P\colon\ I(TT) \overset{m}{=} I(tt) + in + I(re) \wedge NIT(re) .$$

```
tt := TT; re := 0; in := 0;
{invariant P}
while tt ≠ 0 do
   let t be a term in tt with coefficient λ;
   tt := tt - λt;
   if NIT(t) then re := re + λt {P}
   else
      compute s such that (6.3) holds;
      in := in + λ/(m+1) aj^(m+1)_((h-1)x) s ;
      tt := tt - λ/(m+1) aj^(m+1)_((h-1)x) D(s) {P}
   fi
   {P}
od {P ∧ tt = 0, so I(TT) = in + I(re) ∧ 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 *Int* and *Rest*, defined by *Int*(TT) = in and *Rest*(TT) = 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 a1,...,aN. 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 *NIT* and for implementing the mappings *Int* and *Rest* in the MUSIMP system. Moreover, we thank J.M. Kloosterman for writing a MUSIMP implementation of the normalizing mapping *M*.

REFERENCES

Barwise, J. (1977). Handbook of Mathematical Logic, Amsterdam, New York, Oxford: North-Holland Publishing Company.

Ten Eikelder, H.M.M. (1986). Symmetries of the massive Thirring model, J. of Math. Phys. 27, 1404-1410.

Fuchssteiner, B., Oevel, W. and Wiwianka, W. (1987). Computer-algebra Methods for Investigation of Hereditary Operators of higher order Soliton equations, Computer Physics Communications 44, 47-55.

Lang, S. (1965). Algebra, Reading, Massachusetts: Addison-Wesley Publishing Company.

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films
85/04	T. Verhoeff H.M.J.L. Schols	Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate
86/01	R. Koymans	Specifying message passing and real-time systems
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specifications of information systems
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several parallel systems
86/05	Jan L.G. Dietz Kees M. van Hee	A framework for the conceptual modeling of discrete dynamic systems
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers

86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arum Kumar	Compositional semantics for real-time distributed computing (Inf. & Control 1987)
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing
86/11	W.P. de Roever	Questions to Robin Milner - A responders commentary (IFIP86)
86/12	A. Boucher R. Gerth	A timed failures model for extended communicating processes
86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)
86/14	R. Koymans	Specifying passing systems requires extending temporal logic
87/01	R. Gerth	On the existence of a sound and complete axiomatizations of the monitor concept
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases
87/03	G.J. Houben J. Paredaens	A formal approach to distributed information systems
87/04	T. Verhoeff	Delay-insensitive codes - An overview
87/05	R. Kuiper	Enforcing non-determinism via linear time temporal logic specification

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

87/18	C.W.A.M. van Overveld	An integer algorithm for rendering curved surfaces
87/19	A.J. Seebregts	Optimalisering van file allocatie in gedistribueerde database systemen