# Formal Derivation of an Algorithm for Distributed Phase Synchronization

D.S. Buhăceanu[*]    W.H.J. Feijen[†]

September 19, 1996

## Abstract

The main purpose of this paper is to create more evidence for the observation that parallel programs, distributed or not, can be formally — and economically — derived by means of just the predicate calculus and the theory of Owicki and Gries. The example selected here is the problem of Phase Synchronization, in which a number of programs each pass through an unlimited number of phases in a more or less synchronous fashion. A solution is developed for the special case of programs located in the nodes of a tree with communication facilities restricted to communication with neighbouring nodes.

**Keywords and Phrases**: program derivation, multiprogramming, multibounds, theory of Owicki and Gries, predicate calculus, design heuristics, distributed algorithms, phase synchronization.

## 1 Introduction

The most common evaluation of the theory of Owicki and Gries is that it would, at best, be suited for *a posteriori verification* of parallel programs [AO91], and even of only those that cooperate via shared variables. Fortunately, this limited view of the potential of the theory is becoming less and less tenable; by its very simplicity, the theory can pretty well be used for the formal *derivation* of parallel programs.

Over the past few years quite some evidence has been collected that, armed with just the theory of Owicki and Gries and the predicate calculus, one can not only just derive parallel programs but also do it fairly economically. One of the reasons for writing this paper is to show how this works. We develop a *distributed* algorithm in order to demonstrate that the theory of Owicki and Gries is instrumental outside the realm of shared-memory algorithms as well. Of course, one single example cannot exhibit the general mode of deriving parallel programs — multiprograms as we call them — but it can give some of the flavour. More varied accounts can be found in [Moe93] and [vdSom94], and in [Fe90] and [FG95].

## 2 The theory of Owicki and Gries, in brief

In explaining the theory of Owicki and Gries we confine ourselves to what is needed for the understanding of this paper. For more detailed explanations we refer to [OG76] and [Dij82].

A multiprogram is a set of ordinary sequential programs, which we call the (multiprogram's) components. The multiprogram as a whole has a precondition, and the components may be annotated the way we are used to for sequential programs. The theory of Owicki and Gries tells us that this annotation is correct whenever for each assertion it holds that

- it is "locally correct" with respect to the component in which it occurs, i.e. it is established by the (dynamically) preceding atomic statement of that component

[*]Ulenpas 72, 5655 JD Eindhoven, The Netherlands

[†]Dept. of Mathematics and Computing Science, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, The Netherlands; e-mail: wf@win.tue.nl .

- it is "globally correct" with respect to the rest of the system, i.e. it is not falsified by any atomic statement of any other component. (This is usually called "interference freedom".)

In principle, this is all there is to it. Of course, it ought to be clear which statements are atomic. In our forthcoming example we consider assignment statements and so-called guarded skips — statements of the form if $B \rightarrow$ skip fi — to be atomic.

Another important notion in multiprogramming is a system invariant. A relation is a system invariant if and only if it is implied by the multiprogram's precondition and it is not falsified by any atomic statement of any component. As a result, an invariant can be added as a conjunct to *each* assertion, so we can afford the freedom of writing it *nowhere* in the annotation. System invariants greatly contribute to the economy of proving and designing, and, when used in the way indicated above, also to clarity of exposition and economy of writing ... and reading.

Finally, we point out that, in this paper, the guarded skip is our only tool for achieving synchronization. Operationally, guarded skip if $B \rightarrow$ skip fi is equivalent to do $\neg B \rightarrow$ skip od. We handle it via a proof rule, which in Hoare-triple semantics for partial correctness reads

$$\{B \Rightarrow R\}\text{if } B \rightarrow \text{skip fi}\{R\} \ .$$

Its most frequent application is

$$\text{if } B \rightarrow \text{skip fi}\{B\} \ ,$$

for establishing the local correctness of assertion $B$.

## 3 Total deadlock and the multibound

Apart from our concern for partial correctness of a multiprogram, which we capture by providing a correct annotation, we are also confronted with the totally different problem of "individual progress". A component can become blocked indefinitely when it is engaged in executing a guarded skip whose guard never becomes stably true (due to the operations of the rest of the system). Proving individual progress is, in general, quite a nasty task, so nasty even that computing science has not yet succeeded in solving this problem in a technically satisfactory manner.

Under some special circumstances however, the problem of individual progress can be tackled gracefully. Consider, for instance, the following three-component multiprogram, which, projected on the variables $x$, $y$, and $z$, has the form[1]

$$*[\ x{:=}1{+}x\ ]\ ,\ *[\ y{:=}1{+}y\ ]\ ,\ *[\ z{:=}1{+}z\ ]\ .$$

Moreover, suppose that this system maintains invariant

$$MB:\quad x \le K{+}y \wedge y \le L{+}z \wedge z \le M{+}x,$$

for some constants $K$, $L$, and $M$. Then, if one of the components comes to a definite halt so will the others. Consequently, the multiprogram as a whole can display just two scenarios as far as progress is concerned, to wit

- either *all* components get stuck forever — this is called "total deadlock"—
- or *each* component makes progress.

In the presence of a "multibound" like $MB$ we can demonstrate individual progress by showing the absence of the danger of total deadlock. And the nice thing about this is that the latter can be done by just the theory of Owicki and Gries.

Typically, we prove absence of the danger of total deadlock as follows. We insert a correct preassertion to each guarded skip in each component, and we then show that for each combination of guarded skips — taking one from each component — the disjunction of the guards is implied by the conjunction of the corresponding preassertions, cf. [Hoog86].

## 4 The problem of phase synchronization

### 4.1 Specification

The problem of phase synchronization, which we learned from J. Misra [M91], is as follows. We consider an arbitrary, nonempty set of components of the form

[1] $*[T]$ is short for do true $\rightarrow$ $T$ od

$p: \ *[\ S{\cdot}p\ ]$ .

We assume that all invocations of $S{\cdot}p$ terminate, for each $p$. The successive $S{\cdot}p$'s are the successive *phases* of component $p$. The problem now is to synchronize the components in such a way that, when a component is about to start execution of its $(n{+}1)^{st}$ phase, all other components have completed at least $n$ of their phases. In order to render this synchronization requirement more formally and more precisely, we introduce a *fresh* variable $x{\cdot}p$, for each $p$, that keeps track of the number of phases component $p$ has completed. Our first version of the multiprogram thus gets the form

```
Pre:   $\langle \forall q :: x{\cdot}q = 0 \rangle$
-----------------------------------------
$p$ : *[   $\{\langle \forall q :: x{\cdot}p \leq x{\cdot}q \rangle,\ ?\}$
           $S{\cdot}p$
         ; $x{\cdot}p := 1{+}x{\cdot}p$
       ]
```

*version 0*

This, now, is our formal specification. The task ahead of us is to superimpose additional code on the above multiprogram so as to achieve that the plugged-in assertion $\langle \forall q :: x{\cdot}p \leq x{\cdot}q \rangle$ be a correct preassertion to $S{\cdot}p$. We use symbol ? —throughout the paper— to explicitly indicate what remains to be done.
Before embarking on a solution, we observe that, no matter how we proceed in establishing the target assertion $\langle \forall q :: x{\cdot}p \leq x{\cdot}q \rangle$, relation

$$MB : \quad \langle \forall p :: \langle \forall q :: x{\cdot}p \leq 1{+}x{\cdot}q \rangle \rangle$$

will be a system invariant. But this is a perfect multibound! Consequently, to show individual progress in our (ultimate) solution, it suffices to show the absence of total deadlock.

## 4.2 Toward a distributed solution

In version 0 above, the local correctness of target assertion $\langle \forall q :: x{\cdot}p \leq x{\cdot}q \rangle$ is most easily established by a guarded skip with that assertion as a guard. However, this would require facilities for direct information exchange between any pair of components. Our aim, though, is to develop an algorithm for components that form the nodes of a tree and communicate only with components in neighbouring nodes. As a first step towards such a distributed algorithm, we rewrite our target assertion into the equivalent[2]

$$x{\cdot}p \leq \langle \downarrow q :: x{\cdot}q \rangle \ ,$$

and then decide to nominate one fixed component $R$ —to reside in the root of the tree— to keep track of the minimal $x$-value, i.e. to maintain

$$x{\cdot}R = \langle \downarrow q :: x{\cdot}q \rangle \ .$$

This allows us to to rewrite our target assertion into the equivalent

$$x{\cdot}p \leq x{\cdot}R \ .$$

To ensure that the minimal $x$-value will indeed be in the root, we decide that each path from a node towards the root will exhibit a descending sequence of $x$-values. (Thus, the $x$-values form a so-called down-heap.) More precisely, with $f{\cdot}q$ denoting the father of $q$ (for $q \neq R$), we decide to maintain invariant

$$P_0 : \langle \forall q : q \neq R : x{\cdot}(f{\cdot}q) \leq x{\cdot}q \rangle \ .$$

Thus, the next version of our multiprogram becomes —symbol ♡ is used to administer what has been established—

```
Pre:   $\langle \forall q :: x{\cdot}q = 0 \rangle$
Inv:   $MB$, ♡
       $P_0$, ?
-----------------------------------------
$p$ : *[   $\{x{\cdot}p \leq x{\cdot}R,\ ?\}$
           $S{\cdot}p$
         ; $x{\cdot}p := 1{+}x{\cdot}p$
       ]
```

*version 1*

## 4.3 The core of the design

In version 1 above, the local correctness of assertion $x{\cdot}p \leq x{\cdot}R$ could again be easily established by a guarded skip with that assertion as a guard. Although the situation has improved compared to that of version 0, such a guarded skip would still require facilities for direct information exchange between root $R$ and any

[2] $\downarrow$ denotes the minimum

other component. So we must find a different way to guarantee the correctness of $x{\cdot}p \le x{\cdot}R$ . One opportunity is created by exploiting the transitivity of $\le$, viz. by replacing the target assertion by the stronger

$$x{\cdot}p \le \mathit{something} \wedge \mathit{something} \le x{\cdot}R \ .$$

If this is going to work out well, the *something* ought to be an expression that depends only on information of component $p$ and its immediate neighbourhood. In that case the first conjunct becomes a suitable candidate for a guard, and the second conjunct —,and there will be no cure!— had better follow from a system invariant. (In passing, we wish to mention that this strategy is quite general in multiprogramming.) Unfortunately, the conjunct $\mathit{something} \le x{\cdot}R$ cannot follow from one of our current invariants $MB$ or $P_0$, because neither grants us an inequality with $x{\cdot}R$ at the greater side and a *something* of the desired shape at the smaller side. Therefore, an essentially new ingredient has to enter the game.

That new ingredient consists of yet another set of fresh variables, one per component. They serve to eliminate our target assertion $x{\cdot}p \le x{\cdot}R$ in the following way:

$$\begin{array}{ll} & x{\cdot}p \le x{\cdot}R \\ \Leftarrow & \{ \text{ for } \mathit{something} \text{ we choose } y{\cdot}p \ \} \\ & x{\cdot}p \le y{\cdot}p \wedge y{\cdot}p \le x{\cdot}R \\ \Leftarrow & \{ \text{ separate } x\text{'s and } y\text{'s } \} \\ & x{\cdot}p \le y{\cdot}p \wedge y{\cdot}p \le y{\cdot}R \wedge y{\cdot}R \le x{\cdot}R \end{array}$$

Here we decide that the last two conjuncts will follow from new system invariants. The invariance of the middle conjunct requires that the maximal $y$-value reside in the root, and we will meet this requirement by seeing to it that each path from a node to the root will exhibit an ascending sequence of $y$-values. (Thus, the $y$-values form an up-heap.) More precisely, we will maintain as a system invariant

$$P_1 : \quad \langle \forall q : q \ne R : y{\cdot}q \le y{\cdot}(f{\cdot}q) \rangle \ .$$

Adoption of the third conjunct $y{\cdot}R \le x{\cdot}R$ yields, in combination with $P_0$ and $P_1$, the stronger invariant

$$P_2 : \quad \langle \forall q :: y{\cdot}q \le x{\cdot}q \rangle \ .$$

In view of our target assertion $x{\cdot}p \le y{\cdot}p$, it will be necessary to increase $y$'s, and in our next version we immediately do justice to this, and to $P_2$, by plugging in assignments to $y$ so that $P_2$ is satisfied[3]. We obtain

```
Pre:  $\langle \forall q :: x{\cdot}q = 0 \rangle \wedge \langle \forall q :: y{\cdot}q = 0 \rangle$
Inv:  $MB$ , ♡
      $P_0 : \langle \forall q : q \ne R : x{\cdot}(f{\cdot}q) \le x{\cdot}q \rangle$ , ?
      $P_1 : \langle \forall q : q \ne R : y{\cdot}q \le y{\cdot}(f{\cdot}q) \rangle$ , ?
      $P_2 : \langle \forall q :: y{\cdot}q \le x{\cdot}q \rangle$ , ♡
-----------------------------------------------
$p$ : *[  $\{x{\cdot}p \le y{\cdot}p$ , ♡$\}$
          $S{\cdot}p$
         ; $x{\cdot}p := 1 + x{\cdot}p$
         ; $y{\cdot}p := 1 + y{\cdot}p$
       ]
```

*version 2*

Meanwhile, the correctness of our new target assertion $x{\cdot}p \le y{\cdot}p$ has been established by construction. What remains is our care for the invariance of $P_0$ and $P_1$.

## 4.4 Solution

As for $P_0$, increments of $x{\cdot}p$ in component $p$ can violate it. The weakest precondition for $x{\cdot}p := 1 + x{\cdot}p$ not to violate $P_0$ is

$$\langle \forall q : q \ne R \wedge p = f{\cdot}q : 1 + x{\cdot}p \le x{\cdot}q \rangle \ ,$$

which —since $p = f{\cdot}q \Rightarrow q \ne R$— simplifies to

$$\langle \forall q : p = f{\cdot}q : 1 + x{\cdot}p \le x{\cdot}q \rangle \ .$$

We plug this in as a preassertion to $x{\cdot}p := 1 + x{\cdot}p$. Its global correctness is for free because other components only increase their $x$-values. We establish its local correctness by a guarded skip. Notice that the evaluation of this guard by component $p$ requires communication only with $p$'s children. Also notice that this condition simplifies to true for leaf components, so that there the corresponding guarded skips can be omitted.

As for $P_1$, increments of $y{\cdot}p$ in component $p$ can violate it. The weakest precondition for $y{\cdot}p := 1 + y{\cdot}p$ not to violate $P_1$ is

$$\langle \forall q : q \ne R \wedge p = q : 1 + y{\cdot}q \le y{\cdot}(f{\cdot}q) \rangle \ ,$$

which for $p \ne R$ simplifies to

[3]It is the order in which $x{\cdot}p$ and $y{\cdot}p$ are increased that caters for $P_2$

$1+y\cdot p \leq y\cdot(f\cdot p)$

and for $p=R$ to true. Again its global correctness is for free, and its local correctness will be established by a guarded skip. We thus arrive at our next and almost final version —please ignore, for the time being the added assertions—.

```
Pre: $\langle \forall q :: x\cdot q=0 \wedge y\cdot q=0 \rangle$
Inv: $MB,\ P_0,\ P_1,\ P_2$, all $\heartsuit$
-------------------------------------------------
For a non-leaf component $p$, $p\neq R$
*[    $S\cdot p$
      $\{x\cdot p=y\cdot p,\ \heartsuit\}$
    ; if $\langle \forall q : p=f\cdot q : 1+x\cdot p \leq x\cdot q \rangle \rightarrow$ skip fi
    ; $x\cdot p := 1+x\cdot p$
      $\{x\cdot p=1+y\cdot p,\ \heartsuit\}$
    ; if $1+y\cdot p \leq y\cdot(f\cdot p) \rightarrow$ skip fi
    ; $y\cdot p := 1+y\cdot p$
 ]
-------------------------------------------------
For a leaf component $p$
*[    $S\cdot p$
    ; $x\cdot p := 1+x\cdot p$
      $\{x\cdot p=1+y\cdot p,\ \heartsuit\}$
    ; if $1+y\cdot p \leq y\cdot(f\cdot p) \rightarrow$ skip fi
    ; $y\cdot p := 1+y\cdot p$
 ]
-------------------------------------------------
For component $R$
*[    $S\cdot R$
      $\{x\cdot R=y\cdot R,\ \heartsuit\}$
    ; if $\langle \forall q : R=f\cdot q : 1+x\cdot R \leq x\cdot q \rangle \rightarrow$ skip fi
    ; $x\cdot R := 1+x\cdot R$
    ; $y\cdot R := 1+y\cdot R$
 ]
```

*version 3*

And we are through, except for demonstrating the absence of total deadlock. The assertions we added have been included for that purpose.

## 4.5 Absence of total deadlock

We show the absence of the danger of total deadlock by assuming that each component is stuck in an if-statement and deriving the validity of false. Most components have two if-statements; the leaves and the root have just one. We now colour the components either black or white by the convention that a component is

- black when stuck in its first if-statement (absent at the leaves), and
- white when stuck in its second if-statement (absent at the root).

Thus, the root is black and all leaves are white. Consequently, there exists a black component with white children only —nice, little proof omitted—. Let $p$ be such a black component. Because it is blocked in its first if-statement, we have

$(i)\ \ \langle \exists q : p=f\cdot q : x\cdot p \geq x\cdot q \rangle \wedge x\cdot p=y\cdot p$ .

The first conjunct is the negation of the guard and the second conjunct is the preassertion we added to this if-statement.
Because all of $p$'s children are white, hence blocked in their second if-statement, we also have

$(ii)\ \ \langle \forall q : p=f\cdot q : y\cdot q \geq y\cdot(f\cdot q) \wedge x\cdot q=1+y\cdot q \rangle$.

The conjunct $x\cdot q=1+y\cdot q$ is the preassertion we added to this if-statement.
Now let $q$ be a witness for $(i)$, i.e.

$(iii)\ \ p=f\cdot q \wedge x\cdot p \geq x\cdot q \wedge x\cdot p=y\cdot p$ .

By instantiating $(ii)$ for this $q$, we find

$(iv)\ \ y\cdot q \geq y\cdot(f\cdot q) \wedge x\cdot q=1+y\cdot q$ .

We now derive the validity of false:

$$\begin{array}{ll} & x\cdot p \\ \geq & \{\ (iii)\ \} \\ & x\cdot q \\ = & \{\ (iv)\ \} \\ & 1+y\cdot q \\ \geq & \{\ (iv)\ \} \\ & 1+y\cdot(f\cdot q) \\ = & \{\ (iii)\ \} \\ & 1+y\cdot p \\ = & \{\ (iii)\ \} \\ & 1+x\cdot p \end{array}$$

So much for the absence of total deadlock. Thanks to the presence of multibound $MB$, individual progress is guaranteed as well.

## 4.6 A final transformation

In principle, our development has come to an end. Yet, we wish to address two more issues that could be relevant in practice. One is the rather coarse-grainedness of, in particular, the atomic guards $\langle \forall q : p = f \cdot q : 1 + x \cdot p \leq x \cdot q \rangle$, and the other is the problem of ever growing integers $x$ and $y$.

The coarse-grained guards are eliminated as follows. Consider a multiprogram in which one of the components contains guarded skips $\textsf{if } B \wedge C \rightarrow \textsf{skip fi}$. Now, there is a (rather unknown) theorem —The Guard Conjunction Lemma— stating that for globally correct $B$ this coarse-grained guarded skip can be replaced with the succession

$$\textsf{if } B \rightarrow \textsf{skip fi} \ ; \ \textsf{if } C \rightarrow \textsf{skip fi}$$

of these two finer-grained guarded skips, without impairing the correctness of the multiprogram. The tedious proof of this (important) lemma is not given here [Hoom93]. For our current example, in which each individual conjunct $1 + x \cdot p \leq x \cdot q$ is globally correct, this means that the conjuncts can be evaluated one by one, and in any order.

We deal with the ever growing integers by eliminating them. For the elimination of $x$'s, we observe that, thanks to the invariance of $MB$, the maximum and the minimum $x$-values differ by at most 1, and that therefore any two $x$-values differ by at most 1. As a result, we can perform a coordinate transformation towards the boolean domain, viz. by introducing booleans c, one per component, such that

$$c \cdot p \equiv c \cdot q \quad \equiv \quad x \cdot p = x \cdot q \ .$$

We can then replace in our program

$1 + x \cdot p \leq x \cdot q$ by $c \cdot p \not\equiv c \cdot q$ and
$x \cdot p := 1 + x \cdot p$ by $c \cdot p := \neg c \cdot p$

The elimination of the $y$'s is pretty much the same, because we can prove the invariance of the assertion $\langle \forall p, q :: y \cdot p \leq 1 + y \cdot q \rangle$ :

$$\begin{array}{ll}
 & y \cdot p \\
\leq & \{ P_1 \} \\
 & y \cdot R \\
\leq & \{ P_2 \} \\
 & x \cdot R \\
\leq & \{ P_0 \} \\
 & x \cdot q \\
\leq & \{ \text{structure of component } p \} \\
 & 1 + y \cdot q
\end{array}$$

We introduce booleans $d$ to replace $y$, and our final program —of which we now just give the raw code— is

```
Pre: ⟨∀p,q :: c·p ≡ c·q ∧ d·p ≡ d·q⟩

For a non-leaf component p, p ≠ R
*[    S·p
     ; if ⟨∀q : p = f·q : c·p ≢ c·q⟩ → skip fi
     ; c·p := ¬c·p
     ; if d·p ≢ d·(f·p) → skip fi
     ; d·p := ¬d·p
 ]

For a leaf component p
*[    S·p
     ; c·p := ¬c·p
     ; if d·p ≢ d·(f·p) → skip fi
     ; d·p := ¬d·p
 ]

For component R
*[    S·R
     ; if ⟨∀q : R = f·q : c·R ≢ c·q⟩ → skip fi
     ; c·R := ¬c·R
     ; d·R := ¬d·R
 ]
```

## 5 Final remarks

It is quite probable that the algorithm we developed is well-known. After the above was done, we recognized that the algorithm could even have been invented by purely operational considerations. After all, one can view it as a huge two-stage handshake protocol. In the first stage the leaf nodes asynchronously send completion signals towards the root. As soon as this wave of completion signals has reached the root, the latter reflects it as a permission signal which then scatters through the tree giving nodes permission to enter their next phase. Etcetera. (This "metaphor" makes clear that the algorithm is even an efficient one: in a reasonable tree the height of the tree is the logarithm of the number of nodes.)

But even if the algoritm were invented on such grounds, the task of proving its correctness would still remain. And this could be extremely difficult. At best, one would reverse the course of a development, but it is far more likely that an a posteriori proof will enter more cumbersome alleys, if not dead ones. Deriving programs from their specification is just much more economical and satisfactory. This is so because program derivation is a very goal-directed activity, in which the program and its correctness proof see the light in a very harmonious fashion.

The main purpose of this paper was to drive home the message that the derivation of not too trivial an algorithm like the Distributed Phase Synchronization is very well feasible with the simple theory of Owicki and Gries as our only tool for reasoning about multiprograms.

**Acknowledgement**
We are grateful to the members of the Eindhoven Tuesday Afternoon Club for their comments on the first draft of this paper. Special thanks go to Rob R. Hoogerwoord and to Netty van Gasteren for their many valuable criticisms. David Gries went far beyond duty by providing so many fine-grained comments on wording and phrasing. The comments of two anonymous referees have been considered with gratitude, and taken into account.

## References

[AO91] Krzysztof R. Apt and Ernst-Rüdiger Olderog. *Verification of Sequential and Concurrent Programs*. Springer, Berlin, 1991

[Dij82] E.W. Dijkstra. A personal summary of the Gries-Owicki theory. In *Selected Writings on Computing: A Personal Perspective*. Springer, New York, 1982.

[Fe90] W.H.J. Feijen. A little exercise in deriving multiprograms. In W.H.J. Feijen and A.J.M. van Gasteren, D. Gries and J. Misra, editors. *Beauty is Our Business, A Birthday Salute to Edsger W. Dijkstra*. Springer, New York, 1990, pp. 119–127.

[FG95] W.H.J. Feijen and A.J.M. van Gasteren. On multiprogramming. To appear in M. Hinchey and N.Dean, editors.*Educational Issues of Formal Methods*. Academic Press, 1996.

[Hoog86] R.R. Hoogerwoord. An implementation of mutual inclusion. *Information Processing Letters* **23** (1986), pp. 77–80.

[Hoom93] J. Hooman. Properties of program transformations. Technical note, Eindhoven University of Technology ,1993.

[M91] J. Misra. Phase Synchronization. *Information Processing Letters* **38** (1991), pp. 101–105.

[Moe93] P.D. Moerland. *Exercises in Multiprogramming*. Computing Science Notes 93/07, Eindhoven University of Technology ,1993.

[OG76] S. Owicki, D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica* **6** (1976), pp. 319–340.

[vdSom94] F.W. van der Sommen. *Multiprogram Derivations*. Master's thesis, Eindhoven University of Technology ,1994.