From mark Mon Jun 5 12:59:59 1989
To: jtu@hgrrug5.bitnet, wsintom@heitue5.bitnet
Subject: further thoughts regarding your letter

Dear Jan and Tom,

I have now had time to digest your remarks about D being derivable from F-D, the quiescent traces of a process. This fact had indeed escaped us and it leads to a remarkable simplification of the model. I think it will now be possible to distinguish between infinite and unbounded sequences of outputs and to use Roscoe's definedness order as the cpo for recursion.

A process can now be modelled by a set Q of quiescent traces. The set pre(Q) of prefixes of Q can be thought of as the nondivergent traces of Q. There are two closure conditions:

1. Q is closed under reordering.
2. Any nondivergent trace can be extended to a quiescent trace using only output events.

(In your model (Udding's thesis) you only deal with safety and so can make do with specifying the nondivergent traces rather than the quiescent ones. Axioms R3 and R4'' seem to capture reordering. Axiom R2 insists that certain traces are divergences.)

The divergences div(Q) of Q can be derived from pre(Q) as in your letter, except that everything is a divergence when Q is empty!

Q1 is ~~more~~ less defined than Q2 exactly when Q1 = Q2 - div(Q1).

The main advantage of using quiescent traces can be seen in the definition of parallel composition. Both processes have to be quiescent themselves: thus t is a quiescent trace of Q1||Q2 exactly when t reorders some t' which is a quiescent trace of both Q1 and Q2 when restricted to their respective alphabets.

The definitions of nondeterministic choice and guarded choice still need to mention the divergences as well as the quiescent traces of their components. These operators (and parallel composition) can be generalised to take infinite sets of components.

In light of the above, the paper will need a lot of revising! Thank you very much for helping us to simplify things.

Mark Josephs

From mark Fri Jun 2 17:24:36 1989
To: jtu@hgrrug5.bitnet, wsintom@heitue5.bitnet
Subject: Your letter May 29

Dear Jan and Tom,

Thank you very much for your detailed response to our paper, which I received today.

By a remarkable coincidence I have recently turned my attention to the design of delay-insensitive circuits: first studying Alain Martin's method and this week (!) looking at the two theses by Udding and Ebergen. I had noticed that in Chapter 2 of your thesis you present some axioms concerned with reordering. I will certainly give you credit, along with Chandy and Misra, when it comes to revising our paper. Tony and I have been playing with the idea of using Chaos to describe interference in delay-insensitive circuits, so the example on page 3 of your letter accords very much with our current thoughts.

The lack of proofs in our paper is indeed unfortunate. Most importantly, all the operator definitions should yield processes that satisfy the necessary conditions, and all the laws should be correct with respect to those definitions. I had checked this to the best of my ability, and Jifeng had double-checked this. The definition of parallel composition caused me the most difficulty and, as you have kindly pointed out, it still isn't correct. I am not sure that we can do without reordering here - traces closed under reordering may no longer be closed when what was an input channel is turned into an output channel! Fortunately the "self-evident" nature of our laws has meant that they still hold even when we have had to modify the definitions. Jifeng has been working on deriving our laws directly from those of CSP, but this second joint paper is still in preliminary draft form.

Thank you very much again for your interest in our work. I will give careful consideration to all your comments in revising the paper.

I have applied to attend your conference on Mathematics of Program Construction (though I have yet to receive confirmation from you that I will be permitted to attend). I hope this will provide us with a good opportunity to meet and discuss our mutual research interests.

I look forward to seeing you both soon. Best wishes,

Mark

University of Groningen Faculty of Mathematics and Physical Sciences

Department of Computing Science

Dr. Mark B. Josephs
Oxford University Computing Laboratory
Programming Research Group
11 Keble Road
Oxford OX1 3QD
United Kingdom

P.O.Box 800
9700 AV Groningen
The Netherlands
Telephone +3150633939 (secretary)
+3150633943 (extension)

May 29, 1989.

Dear Mark,

We have thoroughly read your paper entitled "A Theory of Asynchronous Processes", coauthored with C. A. R. Hoare and He Jifeng, and we were quite impressed. Through this letter we want to make you aware of our work, the results of which are similar but which builds the mathematical framework in quite a different way. Moreover, we take this opportunity to communicate to you a number of comments upon the paper's contents. For your reference we enclose some relevant literature.

Our work was inspired by Molnar et al. [4], who noticed the need for an asynchronous design methodology for digital electrical circuits. In [8], for the first time as far as we know, the space of delay-insensitive specifications was defined and classified in an axiomatic way. An excerpt of it has appeared as [9]. Other definitions were given later in [7,12,2]. Especially in [12] a number of useful characterizations of that space were given, in particular the one using the notion of convexity. In all this work you find the concept of reordering of traces, which you attribute to Misra. Our metaphor at the time was that of a foam rubber wrapper: even when wrapped in foam, which captures the delays to be unknown and variable, a module should operate correctly. In the next two paragraphs we relate two of your definitions directly to this work.

First of all, we have an infix composability operator $\mathbf{C}$, which takes two traces as arguments. With $s\,\mathbf{C}\,t$ we mean that if s is a sequence of events engaged in by a module then t can be a sequence of events engaged in by that module's environment. Differences between s and t are solely accounted for by the delays in signal transmission between module and environment. (The lengths of s and t may differ due to signals being on their way). Since we have not (yet) been interested in an abstraction level at which we can communicate

values, we have not introduced channels and values communicated through them. Moreover, in our original work we were only interested in safety. In order to compare our two models we make the following assumptions. There is only one value to be communicated, which means that we don't have to tag a communication on a channel with the value communicated, and the set of failures is prefix-closed, meaning that $F = \widehat{F}$ in your terminology. There are ways to introduce values and channels in our model, but the above correspondence suffices for the sake of argument. Now it is easy to prove that your $s \sqsubseteq t$ is our $t \subset s \wedge s\# = t\#$ where $s\#$ denotes the bag of symbols occurring in trace s.

The second interesting thing to notice is that the trace sets of the form $F - D$, where (F, D) is an asynchronous process, can be proved to span the space of delay-insensitive specifications as defined in [8], dropping the requirement of absence of transmission interference (condition R_2). This is most easily shown by using the notion of convexity of [12]. As a matter of fact, F and D can be reconstructed from $F - D$. Given $S = F - D$ then $D = (S^\frown In\_Com^\frown(Com^*)) - S$ and $F = S \cup D$. Also Dill [1] uses pairs of sets to denote asynchronous specifications. A self-contained report on the space of delay-insensitive specifications, linking several of its characterizations has recently appeared as [11].

$S \neq \phi$

In the meantime we have given a much better underpinning of the space of delay-insensitive specifications in [10] and more generally in [0]. The section on alternative representations in [10] and the above remarks should give sufficient clues as to how to transform our model into yours and vice versa. Starting from networks of processes with a formal operational semantics we use so-called testing environments to distinguish these networks with respect to a certain correctness concern. Some networks cannot be distinguished in this way and they are equivalent with respect to that correctness concern. When we take this correctness concern to be absence of computation interference, the resulting space of equivalence classes turns out to be isomorphic to the space of delay-insensitive specifications. In [0] we formally show that for each network an equivalent single-process network exists. The space of processes with a fixed alphabet turns out to be a complete lattice and parallel composition can be defined in terms of the least upper bound of a collection of suitable environments. We also prove a quite useful factorization theorem. This concludes the remarks on the similarities between the two approaches. Because of our different points of departure, there are also a number of differences to observe, which we explore in the next three paragraphs.

It is most interesting to notice that your (informal) *operational* model differs from ours, but that the resulting *denotational* models are isomorphic, as argued above. The need for the introduction of divergence in your model stems

from progress as correctness criterion. Under this criterion infinite chatter is undesired behavior, which comes about *only* by the parallel composition operator. In our model computation interference is considered undesired, since we do not want that to happen in electrical circuits. As a result even the simplest building block has lots of possibilities for computation interference. Nevertheless, the denotational models are isomorphic because they abstract from the operational *cause* of the undesired behavior. In that light the name divergence reminds of one particular operational origin too much to be used in a denotational model, cf. the less biased notion of undefinedness in [6] and your remark on p.7, l.-3.

In [8] we have used a rudimentary language, based upon regular expressions, for circuit specification. There is an obvious syntactic relationship between this language and your process algebra. The meanings of syntactically related terms assigned by the denotational models, however, may differ. Our desire to design electrical circuits calls for a concise description of certain primitive specifications. These specifications have intrinsically many possibilities for undesired behavior. For example, in addition to *Stop* and *Chaos*, the so-called ε-process is quite important for us. It is the process that won't do anything but which diverges as soon as it receives something. We exemplify this further using the specification of a wire.

In our language a wire could be specified by the term $(a;p)^*$, where a is an input and p is an output. The interpretation of it is that a component thus specified can produce a p after having received an a but, in order for this to happen, the environment should wait for p before sending a next a. A specification is meant to prescribe the behaviors of both a component and its environment. Thus, its trace set $(F - D)$ equals the prefix-closure of $\{(ap)^n | n \geq 0\}$. (As said earlier we confine ourselves to safety here and as a result our trace sets are prefix-closed.) The syntactically related term in your algebra is $\mu X.a?;p!;X$. Its meaning in your model, however, is

$$(\{t \mid (\forall s : s \leq t : s\#a \geq s\#p)\}, \emptyset).$$

A term in your algebra that would have the meaning of our wire, is

$$\mu X.a?;[a? \rightarrow \mathit{Chaos} \,\Box\, \mathit{skip} \rightarrow p!;X].$$

Although originally only interested in safety, we are currently also investigating a more general model in order to deal with progress in addition to safety. We also have noticed, as you have, that the refusal sets can almost be eliminated. We start from the same collection of networks as mentioned above to which we add a very weak progress condition for each process. Then we *derive* a denotational model from it using testing equivalence. Unfortunately, we do

not have anything in a definite form yet. This is mainly Tom's work and it will be incorporated in his doctoral thesis which is to appear in September of this year. With the same progress condition, but with a more stringent form of testing, we claim that we can also deal with certain forms of liveness and fairness, but it is too early to substantiate this claim. Apart from this we try to find a (finite) number of primitive delay-insensitive elements which can be proved to be sufficient to implement any delay-insensitive specification, cf. Parrow's work in [5]. With this in mind we are currently trying to classify certain types of non-determinism.

We hope that this gives you an opening to what we consider relevant work in the context of your paper. As for the more detailed comments on your paper we have the following. Our comments fall into two broad categories: general and technical. We first deal with the general remarks.

We find the severest shortcoming of the paper its lack of proofs or references to literature where proofs can be found. Especially the last sentence of your Conclusion raises suspicion:

> Because all our operators have been defined in terms of the model, it *is possible to verify* that they do in fact possess their stated algebraic laws.

We would be most sincerely interested in anything that has been written down to support all the claims. On the other hand, we wish to express our admiration for conveying the intention of your work in Sections 1 and 2 without drowning in technicalities and proofs. As a second point, we find the link with JSD that is announced in the abstract very weak. This may be due to our unfamiliarity with JSD, but the connection might be relativized as far as we are concerned (that way you don't scare potential readers like us either). Instead, it might have been emphasized that a good background in CSP is desirable, not just for Section 2.7 (e.g., the business of alphabets and variables mentioned below in more detail). Thirdly, the lack of a formal operational semantics leaves something to be desired. And, finally, the abundance of laws raises the question as to what guidelines you have employed to select some and omit others. For instance, what to make of the last sentence but one of your Conclusion:

> Indeed, sufficient laws have been provided to permit the transformation of any network of processes into a single sequential process.

Should the reader, at this point, be able to substantiate this claim or will it be the topic of future publications?

Thus far our general comments. Now follow more technical issues. We will refer to lines of the text by page and line number (positive: from the top; negative: from the bottom) based on your 26-page printout of February 11, 1989.

[p.2, l.-14; p.14, l.-10] What brand of fairness do you mean on p.2? Doesn't the remark on p.14 (that X5 expresses a fair merge) contradict the remark on p.2 (that fairness cannot be dealt with)? There are two common usages of the word "fair", one having to do with (repeated) choice and the other with liveness. We consider the first usage to be appropriate, and the second an abuse. We suspect that on p.14 you mean the liveness version and we would suggest that you say "*live* merge" instead. This would also resolve the apparent contradiction.

[p.3, l.-7; p.8, l.11] On p.3 it is stated that *Com* is the alphabet of a process *in the sense of CSP*. What is the alphabet of an asynchronous process? Is it a set of channel names or a set of communication events? Presumably, it is the former (cf. p.19). Therefore, we suggest to add on p.8 the channel set (and its partition into inputs and outputs) as part of the characterization of an asynchronous process. Is it the case that $F \subseteq Com^*$ holds for asynchronous processes and wouldn't this be an additional condition next to (1)–(7)? In general, one should be careful about alphabets: see, for instance, remark below [p.19, l.12] about the trace set T for parallel composition.

[p.4, l.9; p.8, l.12] The condition $D \subseteq F$ should probably be added to the conditions (1)–(7), since it is not implied by them.

[p.4, ll.10–15; p.6, l.-7; p.7, l.-7; p.19, l.-16; p.24, l.8] The informal operational motivation you give for enforcing divergence on unbounded output sequences is weak. The references on p.6 and p.24 to this choice as an "observation" are misleading. Of course, there are good reasons for your choice, but it is just a choice. For example, the relationship with CSP as explained in Section 2.7 immediately gives rise to it. But, apparently, you do not wish to take this relationship with CSP as foundation for your theory.

[p.5, l.-10] You should elaborate: "on distinct channels *of the same direction*". For, otherwise, you might as well leave out the preceding remark about moving inputs before outputs (these occur necessarily on distinct channels). And, furthermore, it would also allow one to move outputs before inputs, which seems undesirable. A nice property to mention here, and for which we will have an application later on (besides Proposition 1, viz. [p.19, l.-7]), is

$$s \sqsubseteq t \Rightarrow s\lceil A \sqsubseteq t\lceil A.$$

It would also be a good idea to point out that $\sqsubseteq$ depends on how the alphabet is partitioned into inputs and outputs. Usually the alphabets are fixed in the

context and it is clear which partitioning is meant, but when defining parallel composition in Section 3.6 there are *three* partitions that play a role.

[p.5, l.-1] The conjunct $F \neq \emptyset$ is not covered by the preceding text, since it has nothing to do with $\sqsubseteq$-closedness. We would suggest to leave it out here and to add $\langle\rangle \in \widehat{F}$ as separate condition later on, which together with (3) yields $F \neq \emptyset$. This separate condition is also more convenient when carrying out proofs, we believe.

[p.6, l.-11] You explain $\lceil$ as denoting restriction of traces (a few lines above), but not $\leq$ as denoting "is a prefix of" for traces.

[p.6, l.-10] It is not obvious why you need to consider that particular case at all and also not why you need not consider any other cases. Clearly, we are missing something. Is it possible that this has to do with unuttered preoccupations about infinite traces?

[Section 2.5] In a sense, the set of divergences is superfluous by conditions (4), (5), and (7), since it can be reconstructed from $F - D$. Hence, also F can be reconstructed from $F - D$, viz. as $(F - D) \cup D$. It might, therefore, be interesting to express all conditions in terms of $F - D$ and its prefix-closure $\widehat{F} - D$. This has already been touched upon in the first part of our letter.

[p.7, l.-5] We already mentioned the weakness of the motivation for (6), which says that if trace $s \in \widehat{F}$ has unbounded output extensions in $\widehat{F}$ then $s \in D$ as well. For instance, what is the "real" reason to require that the left-hand side implies $s \in D$ instead of, for example, $s \in F$. If it is just for the sake of mathematical aesthetics, then what status would you attribute to an operational semantics that does not entail it? It is also not clear why you would require divergences even when there are no infinite computations, for example, for the process with as failure set the prefix-closure of

$$\{\langle c.n\rangle^n \mid n \geq 0\}.$$

[p.8, l.-9] If without this condition—let us call it (8)—refinement is not a *complete* partial order, then what can we make of the remark about recursion on p.10, l.5. Wouldn't (8) be necessary in order to allow recursion in the first place? Or is it the case that the particular limits called for in the least fixed point construction exist independently of (8)? Condition (8) can be expressed entirely in terms of $\widehat{F} - D$: because of condition (5) it is equivalent to "all traces have a finite output successor set in $\widehat{F} - D$". The conjunction of (6) and (8) can be concisely expressed as

$$s \in \widehat{F} - D \;\Rightarrow\; \{t \in \mathit{Out\_Com}^* \mid s^\frown t \in \widehat{F} - D\} \text{ is finite.}$$

[p.9, l.-5] The condition $Q^* = Q$ very much resembles the Foam Rubber

Wrapper Postulate of Molnar [4].

[p.10, l.-2] It is not in line with the rest of the paper that you introduce, without explanation, value-variables into your process algebra. It would be wise to introduce explicitly a set of value-variable names and to emphasize the distinction between x and v. Furthermore, there are some important binding aspects to value-variables. For example, look at how Milner introduces them in [3].

but distracting

[p.11, l.3] Although this way of defining D' immediately entails that D' satisfies condition (4), it may be more appropriate to define it by

$$D' = \{t_0^\frown\langle a.v\rangle^\frown t_1 \mid v \in V \wedge t_0 \in A^* \wedge t_0^\frown t_1 \in D(v)\},$$

where $A = In\_Com - \{a.w \mid w \in V\}$. This shows much clearer to what extent reordering actually takes place. The proof that D' defined like this satisfies condition (4) now also relies on the fact that all $D(v)$ satisfy condition (4). A similar remark holds for the definition of F'.

[p.12, l.7] D' can also be defined without mentioning $\sqsubseteq$ as:

$$\{t_0^\frown t_1 \mid t_0 \in A^* \cap D \wedge t_0^\frown t_1 \in D\} \cup \{t_0^\frown\langle c.v\rangle^\frown t_1 \mid t_0 \in A^* \wedge t_0^\frown t_1 \in D\}$$

where $A = Com - \{c.w \mid w \in V\}$. Here, the initial action "diffuses" a bit further than under input prefixing, since the set A is larger.

[p.13, l.-14] The fact that you only explicitly mention that under these conditions selection will not be indefinitely delayed, suggests that if, for example, there are no *skip*-guarded processes and there *is* input available, then selection *might* be delayed indefinitely. This is unintended, we think: cf. Laws 11 and 24.

[p.16, ll.6–11] In this example it could be helpful to remark that $\langle b.0\rangle$ is not a failure of the first process, but it is of the second (and a divergence in neither).

[p.16, ll.11–20] Here you run again into the problem of not having explained the value-variables and their binding aspects in process terms. In fact, it is unclear which variables are fixed in the context and which are actually meta-variables. We presume that you intend a and x to be fixed, whereas all the others, i.e. b, y, P, and Q, are (meta-)dummies.

[p.17, l.9] Please draw the attention to the fact that "aftering" is not defined for output channels (just like you notice that concealment is not defined for input channels).

[p.18, l.-6] Isn't it a bit late to notice only at this stage that alphabets have

been assumed equal? By the way, what about alphabet requirements for prefixing? Should we have $c \in \mathit{alphabet}(P)$ in order to construct the process $c!0; P$?

[p.19, l.-9] It is not clear that the word "both" applies to both the consistency (cf. the usage of "both" in l.11 above) and the refusing of output. It might be clearer to write ", and refusing to output *in both*".

[p.19, l.12] We presume that $s \in (Com_1 \cup Com_2)^*$ is to be understood as additional condition for membership of T.

[p.19, l.-7] The definition of F' can be simplified by eliminating relation $\sqsubseteq$: not so

$$F' = D' \cup \{t \in T \mid t\lceil A_1 \in F_1 \wedge t\lceil A_2 \in F_2\},$$

where $A_i = In_i \cup Out_i$. This defines the same failure set because of the property mentioned in connection with [p.5, l.-10] and the fact that both F_1 and F_2 satisfy condition (1).

[p.19, l.-13] The definition of D' can also be simplified, no but should be corrected first. A good example to add after X6 is the following. Let us call it X6.5. Consider processes P_1 and P_2 defined by

$$c!0; \mathit{Stop} \quad \text{and} \quad c?x; \mathit{Chaos}$$

respectively, such that $In_1 = \emptyset = Out_2$ and $Out_1 = \{c\} = In_2$. (By the way, we find your implicit treatment of the alphabets of *Stop* and *Chaos* sloppy; especially in Law 29.) Thus, if we compute the parallel composition of P_1 and P_2, we obtain Out_1 as output alphabet, whereas the input alphabet is empty. Hence, in this case, relation $\sqsubseteq$ boils down to trace equality. Furthermore, we find

$$T = \widehat{F}_1 \cap \widehat{F}_2 = \widehat{F}_1 = \{\langle\rangle, \langle c.0\rangle\}$$

and this set is finite. Question: Is $\langle\rangle$ a divergence of the parallel composition? Answer: No, since it belongs to neither of D_1 and D_2 and also cannot arise because of unbounded output sequences in T, which is finite. But $\langle c.0\rangle$ is a divergence, because it is in D_2. Hence, D' does not satisfy condition (5). If the output alphabet Out_2 of P_2 is artificially extended with output channel d, then all of a sudden $\langle\rangle$ is a divergence, because now T has become unbounded. Of course, in the first case, $\langle\rangle$ should also belong to D'. So the definition should be adapted to back-propagate divergences over outputs properly. The reordering can be eliminated just as with F'. We suggest for D'

$$\{s^\frown u \mid \exists t \in Out^*.\, (s^\frown t)\lceil A_1 \in D_1 \vee (s^\frown t)\lceil A_2 \in D_2\} \cup \{\text{"chattering"}\}.$$

Especially, to support definitions like these the reader should be enabled to take a look at proofs. It is not his task to discover them.

[p.22, l.4] If you drop the requirement $C \subseteq Out$ then Law 37 doesn't need that silly requirement $C_1 \cap C_2 = \emptyset$. This resembles the simplification of Laws claimed for "aftering" when defining $P/a.v$ to be equal to P when $a \notin alphabet(P)$, cf. [p.18, l.9].

This rounds off our list of technical comments. We have enclosed some literature for your information. The following remarks are intended as a guideline. The older work, especially [8,12], now mainly has historic value. Of the more recent work, it could be interesting to look at [10], which gives an operational semantics and investigates closed networks of two processes, and at [0], which studies general networks and has as major results the definition of composition in terms of a least upper bound and a factorization theorem. We draw your attention to a subtle difference between the latter two reports: in [10] process alphabets are sets of channel names, whereas in [0] process alphabets are sets of port names, i.e. channel names *combined with a direction.* Finally, in [11] Jan Tijmen's characterization of delay-insensitivity is revisited in the light of recent developments.

We would like to hear more from you. Please, send us other papers of yours that might be of interest to us in this context.

With kindest regards,

Jan Tijmen Udding
Dept of Math and Comp Sci
Groningen University
P.O. Box 800
9700 AV GRONINGEN
The Netherlands
jtu@guvaxin.uucp or
jtu@hgrrug5.bitnet

Tom Verhoeff
Dept of Math and Comp Sci
Eindhoven Univ of Techn
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
wstomv@eutws1.uucp or
wsintom@heitue5.bitnet

encl. References [0], [8], [10], [11], and [12]

References:

[0] W. Chen, J. T. Udding, and T. Verhoeff. Networks of communicating processes and their (de)-composition. In J. L. A. van de Snepscheut, editor, *The Mathematics of Program Construction*, pages ??–??, Springer-Verlag, 1989.

[1] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits.* PhD thesis, C.S. Dept., Carnegie Mellon Univ., Pittsburgh, PA, Feb. 1988.

[2] J. C. Ebergen. *Translating Programs into Delay-Insensitive Circuits.* PhD thesis, Dept. of Math. and C.S., Eindhoven Univ. of Technology, 1987.

[3] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[4] C. E. Molnar, T. Fang, and F. U. Rosenberger. Synthesis of delay-insensitive modules. In H. Fuchs, editor, *1985 Chapel Hill Conference on Very Large Scale Integration*, pages 67–86, Computer Science Press, 1985.

[5] J. Parrow. *Synchronisation Flow Algebra.* Technical Report ECS-LFCS-87-35, Laboratory for Foundations of Computer Science, Dept. of C. S., Univ. of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, U.K., Aug. 1987.

[6] A. W. Roscoe. *Two Papers on CSP.* Technical Report PRG–67, Oxford Univ. Computing Laboratory, Programming Research Group, 8–11 Keble Road, Oxford OX1 3QD, U.K., July 1988.

[7] H. M. J. L. Schols. *A Formalisation of the Foam Rubber Wrapper Principle.* Master's thesis, Dept. of Math. and C.S., Eindhoven Univ. of Technology, 1985.

[8] J. T. Udding. *Classification and Composition of Delay-Insensitive Circuits.* PhD thesis, Dept. of Math. and C.S., Eindhoven Univ. of Technology, 1984.

[9] J. T. Udding. A formal model for defining and classifying delay-insensitive circuits. *Distributed Computing*, 1(4):197–204, 1986.

[10] J. T. Udding and T. Verhoeff. *The Mathematics of Directed Specifications.* Technical Report WUCS-88-20, Dept. of C.S., Washington Univ., St. Louis, MO, June 1988.

[11] T. Verhoeff. *Characterizations of Delay-Insensitive Communication Protocols.* Technical Report 89/06, Dept. of Math. and C.S., Eindhoven Univ. of Technology, May 1989.

[12] T. Verhoeff. *Notes on Delay-Insensitivity.* Master's thesis, Dept. of

Math. and C.S., Eindhoven Univ. of Technology, 1985.